COAMPS™
Version 3 Model Description

General Theory and Equations

COUPLED OCEAN/ATMOSPHERE MESOSCALE PREDICTION SYSTEM

Naval Research Laboratory • Marine Meteorology Division
Monterey, California

NRL Publication
NRL/PU/7500--03-448
May 2003

Contents

Introduction

The Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS™)* is the latest product in a series of mesoscale model developments at the Naval Research Laboratory (NRL) Marine Meteorology Division (MMD). COAMPS represents state-of-the-art analysis (including the Nowcast capability) and short-term (up to 72 hours) forecast tools applicable for any given region of the Earth in both the atmosphere and ocean. In the winter of 1996-summer of 1997, the atmospheric data assimilation and prediction components and the sea-surface temperature (SST) analysis component of COAMPS were transitioned to Fleet Numerical Meteorological and Oceanographic Center (FNMOC). In this implementation it was run on a CRAY Y-MP. In early 1997, the next phase involved porting COAMPS to run on smaller workstations such as SGI IRIX, DEC ALPHA, IBM AIX, SUN SOLARIS, HP UX, and personal computers (PCs). In June 1997, NRL MMD successfully demonstrated the Shipboard Tactical Atmospheric Forecast Capability (STAFC) by running COAMPS real-time aboard the USS *Nimitz* on a Hewlett Packard workstation.

In 1998, NRL MMD received the Office of Naval Research (ONR) Blue Book award to deliver COAMPS as part of the Tactical Atmospheric Modeling System-Real Time (TAMS-RT) system. It was delivered to the Naval Pacific Meteorology and Oceanography Center (NAVPACMETOCCEN) at San Diego, California, and the Naval Central Meteorology and Oceanography Center (NCMOS) at Bahrain. TAMS-RT was developed by NRL MMD to provide Navy regional forecast centers a means to obtain higher temporal and spatial forecast resolution by running COAMPS locally on an SGI workstation. It also gave regional centers the flexibility to tailor the modeling system to their needs.

Because of the success in running and using TAMS-RT at these two Navy regional forecast centers, the TAMS-RT system was transitioned to FNMOC in 1999. FNMOC later added its data distribution system, METCAST, to TAMS-RT. In 2000, this integrated system was delivered as the Distributed Atmospheric Mesoscale Prediction System (DAMPS) to other Navy regional forecast centers.

NRL MMD continued improving TAMS-RT and added the Defense Information Infrastructure/Common Operating Environment (DII/COE). This enhancement became the COAMPS-On-Scene (COAMPS-OS) real-time forecasting system. In 2001, it was delivered to the Naval Meteorology and Oceanography Command (CNMOC) as a software segment of the Navy Integrated Tactical Environment System (NITES).

In early 2001, the COAMPS Process Action Team was formed, with members from the Oceanographer of the Navy (NO96) staff; the Commander, CNMOC; ONR; and NRL. They approved the general release of the shared-memory version of COAMPS through the COAMPS web site <http://www.nrlmry.navy.mil/projects/coamps>. COAMPS was made available via the World Wide Web to increase its usage by a broader research community, enabling it to be fully tested over a wide range of phenomena.

In 1998, when the trend of high-performance computing was moving toward massive parallel distributed-memory computer systems, NRL MMD scientists collaborated with scientists from the Lawrence Livermore National Laboratory (LLNL) (Dr. Arthur Mirin and Dr. Gayle Sugiyama) to begin development of a scalable version of COAMPS. This version of COAMPS uses the Message Passing Interface (MPI) and a horizontal domain decomposition technique to achieve the parallelism. It became fully operational at FNMOC on the SGI O3K systems in the winter of 2001. This version of COAMPS includes an atmospheric data assimilation system that is based on a three-dimensional Multivariate Optimum Interpolation (MVOI) method and a nonhydrostatic, multinested atmospheric forecast model.

*COAMPS is a registered trademark of the Naval Research Laboratory.

COAMPS continues to be expanded, and components of COAMPS are currently under further development. An ocean data assimilation system composed of a three-dimensional MVOI ocean analysis and a hydrostatic forecast model is one area of expansion. A three-dimensional atmospheric variational analysis capability, known as the NRL Atmospheric Variational Data Assimilation System (NAVDAS), is another area of development.

Many NRL MMD scientists and COAMPS users have contributed to the development and testing of COAMPS. It is impossible to name them all, but their contributions are greatly acknowledged. This document is a collection of COAMPS module descriptions from several COAMPS developers. It is written with the intention that users gain an understanding of how the COAMPS analysis and forecast models work. The general theory and equations, including journal references and naming conventions of the equations used in the source code, are discussed. Common user-specified namelist variables and an example of how to use them are provided. Source code flow charts are presented, where applicable, to aid the discussion.

acknowledgments

Acknowledgments

This document is the combined effort of several people affiliated with the Naval Research Laboratory in Monterey, CA. The content for the chapters was provided by the following researchers (in alphabetic order): Sue Chen, James Cummings, James Doyle, Richard Hodur, Teddy Holt, Chi-Sann Liou, Ming Liu, Arthur Mirin, James Ridout, Jerome Schmidt, Gayle Sugiyama, and William Thompson. Sarah Bargsten of Computer Sciences Corporation provided technical editing.

The support for COAMPS development, provided by grants obtained from the Office of Naval Research through programs PE-0602435N and PE-0602704N, the Naval Research Laboratory through program PE-0601153N, and Space and Naval Warfare Systems Command through program PE 0603207N is gratefully acknowledged. Further information about COAMPS can be accessed on the Internet at: http://www.nrlmry.navy.mil/projects/coamps.

Chapters and contributing authors are as follows:

Chapter 1: Overview (Sue Chen and Richard Hodur)
Chapter 2: COAMPS Atmospheric Analysis (Teddy Holt and Sue Chen)
Chapter 3: Ocean Data Assimilation (James Cummings)
Chapter 4: Description of the COAMPS Atmospheric Model
- 4.1 COAMPS Model Numerics (James Doyle)
- 4.2 Moist Physics (Jerome Schmidt)
- 4.3 Cumulus Scheme (James Ridout)
- 4.4 COAMPS Radiation Module (Chi-Sann Liou)
- 4.5 Surface and Planetary Boundary Layer Parameterization (William Thompson)
- 4.6 COAMPS Message Passing Interface (Teddy Holt, Arthur Mirin, Gayle Sugiyama, Jerome Schmidt, Richard Hodur, and Sue Chen)

Chapter 5: Aerosol-Tracer Module (Ming Liu).

1. Overview

The atmospheric component of COAMPS can be used for real-data or for idealized applications. For the real-data applications, the COAMPS analysis can use either global fields from the Navy Operational Global Atmospheric Prediction System (NOGAPS) or the most recent COAMPS forecast as the first-guess. Observations from aircraft, rawinsondes, ships, and satellites are blended with the first-guess fields to generate the current analysis. For idealized experiments, the initial fields are specified using an analytic function and/or empirical data (such as a single sounding) to study the atmosphere in a more controlled and simplified setting. The atmospheric model uses nested grids to achieve high resolution for a given area; it contains parameterizations for subgrid scale mixing, cumulus parameterization, radiation, and explicit moist physics. Examples of mesoscale phenomena to which COAMPS has been applied include mountain waves, land-sea breezes, terrain-induced circulations, tropical cyclones, mesoscale convective systems, coastal rainbands, and frontal systems.

The COAMPS model domain typically covers a limited area over the Earth. The model grid resolution may range from a few hundred kilometers (synoptic scale) to approximately 100 meters. The actual dimensions applied depend on the scale of phenomena that the user is interested in simulating. The model dimensions can be set to produce any rectilinear pattern. In addition, it can be rotated to align with any surface feature, such as the terrain or a coastline. COAMPS can be run with any number of nested grids, with the requirement that the horizontal grid resolution in any mesh be one-third that of the next coarser mesh.

The COAMPS atmospheric system consists of two major components – analysis and forecast. The COAMPS analysis executable is run first to prepare the initial and boundary files used in the forecast model. The COAMPS forecast executable performs time integration of the model numerics and physics. It then outputs prognostic and diagnostic fields in pressure, sigma, or height coordinates. Options for running the analysis and forecast are specified through several Fortran namelists (detailed in Appendix B).

1.1 Ocean Analysis

The COAMPS ocean analysis is composed of the three-dimensional MVOI method that analyzes ocean observations (such as ships, buoys, and satellites) to provide the atmospheric model with sea surface temperature (SST) and sea ice analysis. In a previous version of COAMPS, the ocean SST and sea ice analysis capability was performed within the atmospheric analysis using a subroutine call from the analysis driver routine *coama*. The ocean analysis is now available as a stand-alone system. The SST and sea ice fields are read in from the atmospheric driver *coama*. In the stand-alone atmospheric forecast model, the SST field remains constant throughout the forecast.

1.2 COAMPS Atmospheric Model Horizontal and Vertical Grids

Both the horizontal and vertical grids in the forecast model are staggered. The horizontal grid uses the Arakawa-Lamb (1977) scheme C staggering depicted in the following diagram. The scalar variables (denoted π) are defined in the center of the model grid box:

$\pi_{i-1\ j+1}$	$u_{i-1\ j+1}$	π_{ij+1}	u_{ij+1}	$\pi_{i+1\ j+1}$
$v_{i-1\ j}$		v_{ij}		$v_{i+1\ j}$
$\pi_{i-1\ j}$	$u_{i-1\ j}$	π_{ij}	u_{ij}	$\pi_{i+1\ j}$
$v_{i-1\ j-1}$		v_{ij-1}		$v_{i+1\ j-1}$
$\pi_{i-1\ j-1}$	$u_{i-1\ j-1}$	π_{ij-1}	u_{ij-1}	$\pi_{i+1\ j-1}$

The *u* component is one half of a grid distance to the right and has the rightmost column padded with zero. The *v* component is one half of a grid distance to the north and has the uppermost row padded with zero. Because of the horizontal grid staggering, averaging is required for computing the gradient terms of model finite-differencing equations and when using scalar variables at the mass points to compute terms used in the *u* and *v* equations or vise versa. Equation (1.1) demonstrates horizontal averaging for the Coriolis term in the *u*-equation of motion for grid point (*i*,*j*):

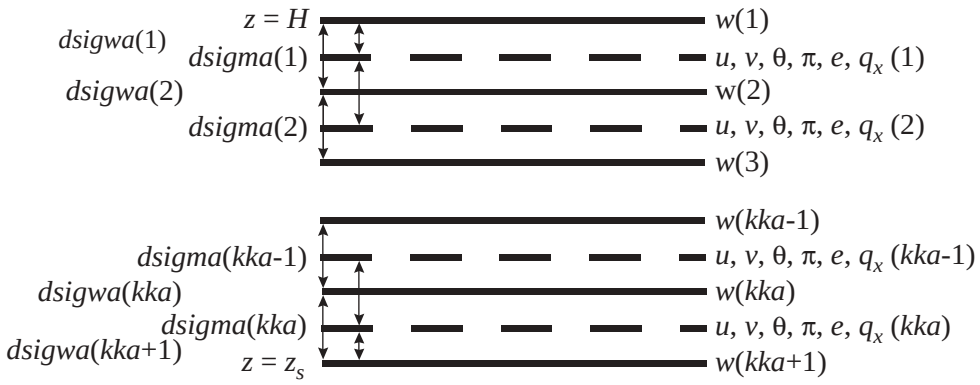
$$fv = \frac{(f_{i+1,j} + f_{i,j})}{2}\left(\frac{v_{i,j} + v_{i+1,j} + v_{i,j-1} + v_{i+1,j-1}}{4}\right). \tag{1.1}$$

In addition, the horizontal domain computation range (Fortran "do loop") within the code is different for the scalar, *u*, and *v* variables (see Chapter 4.6.2.1 for more discussion on the do loop ranges).

The COAMPS analysis is performed on the Arakawa-Lamb scheme A grid (i.e., no grid staggering). The bicubic spline interpolation is used to interpolate the analyzed fields to the C grid within the forecast model code. This is performed in subroutines *stdp2z* and *inlvl*.

The vertical coordinate is a terrain-following sigma *Z* system. The vertical velocity *w* is defined at the model *sigmwa* levels. The model sigma (σ) is defined as

$$\sigma = \frac{H(z\text{-}z_s)}{(H\text{-}z_s)}$$



H corresponds to the depth of the atmosphere and z_s corresponds to the value of the terrain at the grid point.

The namelist input variable *dsigma* defines the layer thickness between successive *w* levels. All other prognostic variables (*u*, *v*, *e*, θ, π, *q*, *qc*, *qr*, and *qs*) are located halfway between two *sigmwa* levels. Selection of the *dsigma* array to define the model atmosphere depends not only on the weather phenomena of interest, but also on the proper representation of the whole troposphere. The number of available vertical sigma levels is limited to 300; in general, 30 *dsigma* levels are appropriate for most NWP applications. If the user is running high horizontal resolution over steep topography, it may be important to increase the vertical resolution as well (e.g., more than 30 sigma levels).

In COAMPS, the vertical indexing is top down (i.e., *k*=1 is the top of the model atmosphere). The scalar variables must be averaged to the *w* levels when scalar variables are used to compute variables defined at the *w* levels. In the code, the vertical weighting variables *dsav*1*a* and *dsav*2*a*, computed by subroutine *atmos/libsrc/asharelib/checkz*, are multiplied by the scalar variables to obtain the averaging value at the *w* levels. This is given by

$$S_w(i,j,k) = S(i,j,k) * dsav1a(k) + S(i,j,k-1) * dsav2a(k). \tag{1.2}$$

The S term represents any variables other than w. S_w represents its value at the w level. The variables *dsav1a* and *dsav2a* are defined by

$$dsav1a(k) = \frac{dsigma(k-1)}{dsigma(k-1) + dsigma(k)} \tag{1.3}$$

and

$$dsav2a(k) = \frac{dsigma(k)}{dsigma(k-1) + dsigma(k)}. \tag{1.4}$$

Table 1.1 provides atmospheric vertical level information.

Table 1.1 — Vertical Levels

k	sigmaw	dsigw
1	31050.000	7500.000
2	24400.000	5800.000
3	19400.000	4200.000
4	16050.000	2500.000
5	14300.000	1000.000
6	13300.000	1000.000
7	12425.000	750.000
8	11675.000	750.000
9	10925.000	750.000
10	10175.000	750.000
11	9425.000	750.000
12	8675.000	750.000
13	7800.000	1000.000
14	6800.000	1000.000
15	5800.000	1000.000
16	4800.000	1000.000
17	3900.000	800.000
18	3100.000	800.000
19	2300.000	800.000
20	1600.000	600.000
21	1100.000	400.000
22	750.000	300.000
23	500.000	200.000
24	330.000	140.000
25	215.000	90.000
26	140.000	60.000
27	90.000	40.000
28	55.000	30.000
29	30.000	20.000
30	10.000	20.000
k	**sigmaw**	**dsigw**
1	34800.000	3750.000
2	27300.000	6650.000
3	21500.000	5000.000
4	17300.000	3350.000
5	14800.000	1750.000
6	13800.000	1000.000
7	12800.000	875.000
8	12050.000	750.000
9	11300.000	750.000
10	10550.000	750.000

Table 1.1 (continued) — Vertical Levels

k	sigmaw	dsigw
11	9800.000	750.000
12	9050.000	750.000
13	8300.000	875.000
14	7300.000	1000.000
15	6300.000	1000.000
16	5300.000	1000.000
17	4300.000	900.000
18	3500.000	800.000
19	2700.000	800.000
20	1900.000	700.000
21	1300.000	500.000
22	900.000	350.000
23	600.000	250.000
24	400.000	170.000
25	260.000	115.000
26	170.000	75.000
27	110.000	50.000
28	70.000	35.000
29	40.000	25.000
30	20.000	20.000
31	0.000	10.000

1.3 Map Projection and Map Scale Factor

COAMPS offers several map projections from which to choose. The map projections create a representation of the Earth's spherical surface on a flat surface. The available projections and their optimal geographic applications are as follows: Polar Stereographic for high latitudes, Lambert Conformal for mid-latitudes, Mercator and Spherical for lower latitudes, and Cartesian for idealized grid.

Some exceptions must be considered when choosing a projection. In general, the observed wind variable must be rotated to the model grid (except for Mercator and Cartesian projections). The rotation accounts for the fact that, for these projections, the *x* and *y* directions in the COAMPS grid do not correspond to the east-west and north-south direction on the Earth. The COAMPS sigma-level output *u* and *v* fields are always relative to the model's grid coordinate. The pressure and the height level *u* and *v* field's output are de-staggered, and also are not rotated back to the Earth's spherical surface.

The grid length usually varies across the domain. Map-scale factor *m* (i.e., distance on the map divided by the actual distance on Earth), is taken into account whenever horizontal gradients are used in the model equations. The map scale factors applied in COAMPS equations are defined as 1/*m* (represented by the array's *hx* and *hy* in the code). Therefore, the actual distance on Earth is equal to the product of the model horizontal grid distance and the map-scale factor. The map-related variables are computed in subroutine *share/libsrc/coampslib/grid.F*.

The model time step is limited by the minimum map scale factors in the Courant-Friedrichs-Levy (CFL) equation within the forecast domain. Because of this restriction, a domain with map scale factors (1/*m*) close to 1.0 at all points will minimize the computational cost by enabling the largest time step to be taken.

2. COAMPS Atmospheric Analysis

The atmospheric analysis driver for COAMPS is a Fortran routine called *atmos_analysis.F.* The purpose of this module is to initialize the environment for the atmospheric analysis. The *atmos_analysis.F* begins by specifying default values for namelist variables, reading the COAMPS namelist, assigning the forecast starting date-time-group (DTG), initializing arrays, and allocating memory for the analysis grids. The final task of *atmos_analysis.F* is to call the COAMPS atmospheric analysis routine *coama.F*.

2.1 Analysis Driver *atmos_analysis.F*

The analysis driver in COAMPS uses namelist variables to control parameter values. Modifying the namelist variables allows the user to create a customized COAMPS analysis.

2.1.1 atmosnl namelist

The *atmosnl* namelist is used in *atmos_analysis.F* to define the model setup. Table 2.1 lists namelist variables for *atmosnl* and briefly describes their purpose and default values. Unless otherwise noted, all namelist variables are integers. Note that the maximum number of nest levels within COAMPS is seven, specified by *mxgrds*.

Table 2.1 — *atmosnl* Namelist

Parameter	Description	Default Value
cdtg	Date-time-group (character*10)	1995010100
iaero	Passive-tracer switch 0 = off 1 = on	0
ianal	Analysis boundary condition switch 0 = compute output boundary tendencies for the coarse domain only 1 = compute analysis and boundary tendencies 2 = compute analysis only 3 = compute Nowcast only	1
idelay	Delay time for nests (hour, minute, second for each nest)	21*3
ldigit	Digital filter switch (logical)	false
lgrdall	Passive tracer grid switch (logical) true = passive tracer for all grids false = passive tracer for specified grids only	true
lnmove	Switch to move nests (logical) true = move nests false = no move	***mxgrds****false
lm	Number of atmospheric analysis pressure levels	16
lmbc	Number of atmospheric boundary condition pressure levels	16

Table 2.1 (continued) — *atmosnl* Namelist

Parameter	Description	Default Value
maxnest	Maximum number of nests	**mxgrds**
mbin	Number of passive tracer bins	1
mspc	Number of passive tracer species	1
nbdya	Boundary condition option used for digital filter	7
nbdypt	Number of grid points used for boundary condition blending	5
nestcc	Grid number for passive tracer option ***lgrdall***=false	1
nbnam	Number of halo points	1
loopvecnam	Loop vectorization option 0 = no vectorization 1 = vectorization	1
ndxnam	Number of domains in x-direction for each nest	1,1,1,1,1,1,1
ndynam	Number of domains in y-direction for each nest	1,1,1,1,1,1,1
npr0nam	Processor number for root processor ***npr0***	0

2.1.2 Domain decomposition

The atmosphere analysis is performed across the entire domain rather than breaking it down into subsectors because the COAMPS analysis is run using OpenMP <http://www.openmp.org> multitasking directives. This is in contrast to the Message Passing Interface (MPI) distributed memory option used in the forecast model. The *domdec_sm_init* module is used to initialize the analysis domain to extend across the full range of the data (i.e., from 1 to *ma* in the x-direction and from 1 to *na* in the y-direction) for the subroutines that are used by both the analysis and forecast models.

2.1.3 Memory allocation

Memory allocation for the atmospheric analysis is handled in the *mema.F* routine. The main model variables needed for each mesh are stored in a single one-dimensional array called *a*(). The *a*() array size is dynamically set at runtime based on the user-defined grid information defined within the namelist. Each model variable in COAMPS is given a starting element number, referred to as a pointer, within *a*(). The pointer is a grid-dependent integer given by the name of the variable preceded by the letter "i". For example, the starting location of a variable named *var1* on grid *nn* is referenced in the *a*() array as *a(ivar1(nn))*.

Each model variable is assigned a given length of the *a*() array, depending on the size of the variable for a particular mesh. The size of each variable stored in *a*() is added sequentially for each mesh to arrive at the total size of *a*() required to run the simulation. Array types are separated into two-dimensional, three-dimensional, and lateral boundary file types. The pointers for each model variable are stored in one-dimensional arrays that comprise the common blocks defined in the Fortran include file *apointers.h.*

2.2 Atmospheric Analysis Routine *coama.F*

The purpose of the atmospheric analysis routine *coama.F* is to produce a set of initial conditions for the atmospheric forecast model. The analysis routine consists of five main components: (1) setup parameters, (2) setup grid, (3) setup surface fields, (4) read and process initial fields, and (5) output. Figure 1 illustrates the flow of the analysis routine.

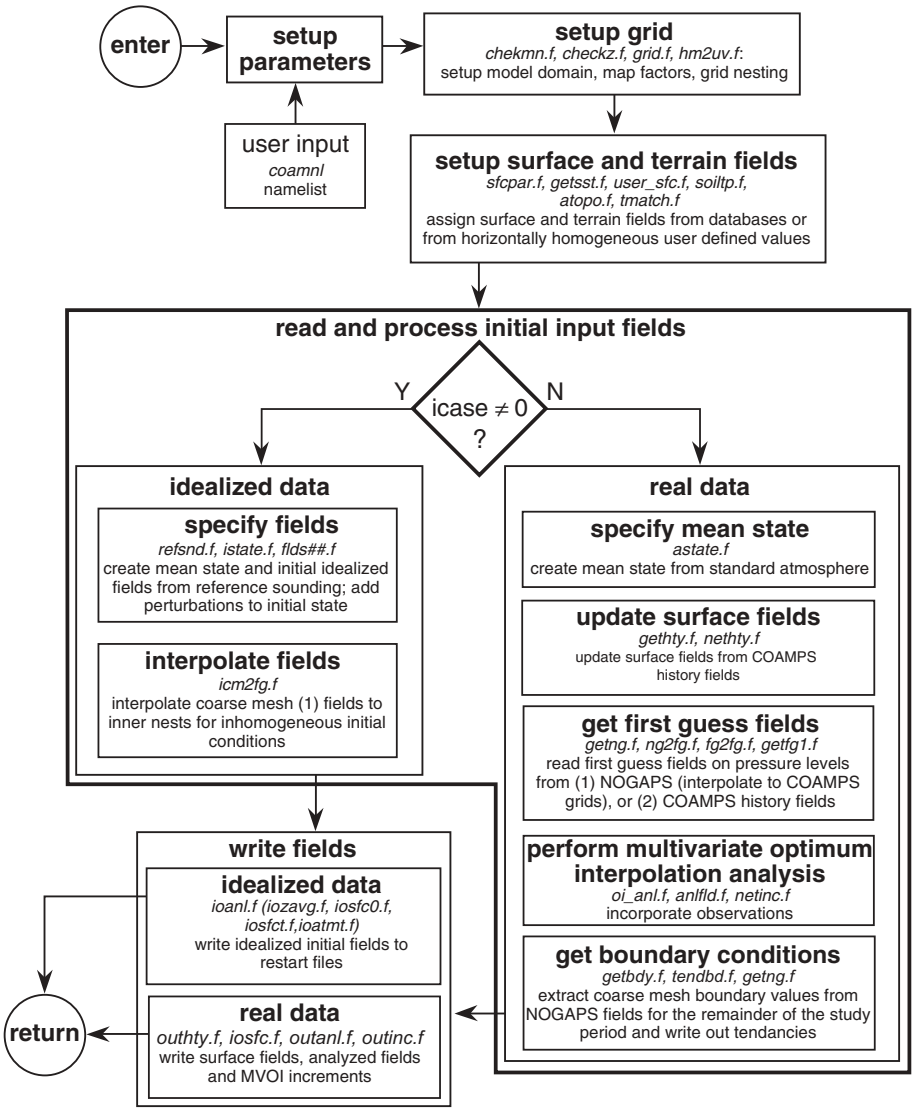


Figure 1
Flow diagram of the analysis routine *coama.f*. The analysis routine includes five main steps: setup parameters, setup grid, setup surface and terrain fields, read and process initial input fields, and write fields. The subroutines called in each step are shown in italics. The atmospheric fields may be either idealized or real data, as indicated by the value of namelist parameter *icase*. (From Haack 1996)

2.2.1 Setup parameters

The analysis also uses namelist variables to control parameter values similar to the atmospheric analysis driver. The primary namelist is *coamnl*.

2.2.1.1 *coamnl* namelist

The COAMPS Software User's Manual (Haack 1996) describes the *coamnl* variables in detail. New variables for *coamnl* are listed in Table 2.2 with a brief description of their purpose and default values. All namelist variables are integers unless otherwise noted. The maximum number of fields is controlled by several parameters in *coamnl.h*: *max1d* = 1000, *maxgrds* = 7, *maxpt1d* = 30, *ilen* = *max1d***maxgrds*, *ilen3* = 3**max1d***maxgrds*.

Table 2.2 — *coamnl* Namelist

Parameter	Description	Default Value
ddata	Path for idealized data (char*80)	‘ ‘
messg	Message (char*32)	‘ ‘
type1k	High-resolution terrain data type (char*32)	DTED1_010
cstline	Coastline parameter (real)	***maxgrds****0.0
hmt	Idealized mountain height (real)	0.0
iwvlng	Silhouette terrain parameter (real)	2.0
nslflt	Silhouette terrain filter parameter	***maxgrds****0
nsrch	Search box size for silhouette terrain	2,2,5,2,1,1,1
prbc	Pressure levels (hPa) for analysis boundary conditions (real)	10.0, 20.0, 30.0, 50.0, 70.0, 100.0, 150.0 200.0, 250.0, 300.0, 400.0, 500.0, 700.0 850.0, 925.0,1000.0,84*0.0
r1dlat	Land-water latitude location parameter (real)	***ilen****0.0
r1dlon	Land-water longitude location parameter (real)	***ilen****0.0
silwgt	Silhouette terrain weight parameter (real)	***maxgrds****1.0
silmax	Silhouette terrain averaging parameter (real)	***maxgrds****1.0
toffset	Idealized offset value (real)	0.0
topores	Terrain horizontal resolution (km) (real)	***maxgrds****1.0
ibiod	Output parameter for surface fields	0
ibog1	Size of pseudo-observations analysis volumes (x-direction)	0
icup	Cumulus parameterization type	1,1,1,1,1,1,1
idbms	I/O parameter type	1
ilandu	Land-use type	***maxgrds****0
ilw1d	Land-water parameter	***ilen****9
ioizi	Zero-increment option for MVOI	***maxgrds****0
iraob1	Threshold number of rawindsonde for each MVOI volume	2
meso_init	Option for interpolating first-guess fields to COAMPS coarse mesh 0 = use NOGAPS as first guess and boundary condition 1 = use COAMPS as first guess and boundary condition 2 = output additional COAMPS flat files to be used as boundary condition for the next meso-init forecast.	0
meso_grid	Grid number for meso_init	1
meso_start	Start time for meso_init	0,0,0
meso_end	End time for meso_init	0,0,0
meso_cycle	Cycle time for meso_init	0,0,0
lm_meso	Number of vertical levels formeso_init	30

Table 2.2 (continued) — *coamnl* Namelist

Parameter	Description	Default Value
pr_meso	Pressure levels (hPa) for meso_init (real)	10.0, 20.0, 30.0, 50.0, 70.0, 100.0, 150.0 200.0, 250.0, 300.0, 350.0, 400.0,450.0 500.0 ,550.0, 600.0, 650.0, 700.0, 750.0 775.0, 800.0, 825.0 ,850.0, 875.0, 900.0, 925.0, 950.0, 975.0, 1000.0, 1013.0,70*0.0
isoiflg	Idealized deep soil temperature flag	***maxgrds****1
itaubc	Starting TAU to search for NOGAPS boundary condition	0
itaung	Time interval to search for NOGAPS boundary condition	6
itaus	Starting time for analysis boundary condition	0,0,0
i1km	High-resolution terrain index	1
jbog1	Size of pseudo-observations analysis volumes (y-direction)	0
jcm2fg	Idealized interpolation parameter	0
ksavdig	Digital filter time parameter	-1,0,0
digtim	Digital filter time parameter (real)	6.0, 0., 0
ksavef	Restart save parameter	-1,0,0
ksaver	Restart save parameter	-1,0,0
ksavpcp	Precipitation bucket frequency parameter	***maxgrds****(72,0,0)
ksavmsp	Maximum wind speed output parameter	-1,0,0
nbio	Surface output field number parameter	1
nbiox	Surface output field number parameter	***maxpt1d****5
nbioy	Surface output field number parameter	***maxpt1d****5
noiexp	Number of MVOI expansions	3
ifsave	Output save parameter	1
isavefrq	Output save frequency parameter	-1,0,0
lshrad	Number of shortwave radiation passes	4
ncast_start	Nowcast start parameter	0
ncast_end	Nowcast end parameter	0
ncast_cycle	Nowcast cycle parameter	0
kmbbeg	Budget begin parameter	99999
kmbend	Budget end parameter	99999
kmbinc	Budget increment parameter	99999
imovtyp	Moving nest type parameter	***maxgrds****1
itmove	Moving nest move times	***ilen3****-1
nmovei	Number of move grid points in x-direction	***ilen****0
nmovej	Number of move grid points in y-direction	***ilen****0
movemx	Maximum number of move grid points per time step	1,3,9,27,27,27,27
xlatmov	Ending latitude for ***imovtyp***=2	***ilen****-999.99
xlonmov	Ending longitude for ***imovtyp***=2	***ilen****-999.99
ioffset	Idealized data offset parameter	0
joffset	Idealized data offset parameter	0
kwidth	Idealized width parameter	0
nwidth	Idealized width parameter	0

Table 2.2 (continued) — *coamnl* Namelist

Parameter	Description	Default Value
lcoare	TOGA COARE surface flux parameter (logical)	false
lcupar	Cumulus parameterization parameter (logical)	true
lpseud	Pseudo-observation parameter (logical)	false
lsilhout	Silhouette terrain parameter (logical)	false
lvisfn	Output visual parameter (logical)	false
lviz5d	Output VIS5D parameter (logical)	false
l1dij	One-dimensional output parameter (logical)	true
l2way	Two-way interaction parameter (logical)	false
l2wayi	Two-way interaction parameter (logical)	true
labdye	Parameter to turn on/off *abdye* (logical)	true
labdytq	Parameter to turn on/off *abdytq* (logical)	true
labdyuv	Parameter to turn on/off *abdyuv* (logical)	true
lafore	Parameter to turn on/off *afore* (logical)	true
laforqx	Parameter to turn on/off *aforqx* (logical)	true
lalhs	Parameter to turn on/off *alhs* (logical)	true
lamixtq	Parameter to turn on/off *amixtq* (logical)	true
lamixnf	Parameter to turn on/off *amixnf* (logical)	true
lamixuv	Parameter to turn on/off *amixuv* (logical)	true
lamixw	Parameter to turn on/off *amixw* (logical)	true
larhsp	Parameter to turn on/off *arhsp* (logical)	true
larhsu	Parameter to turn on/off *arhsu* (logical)	true
larhsv	Parameter to turn on/off *arhsv* (logical)	true
larhsw	Parameter to turn on/off *arhsw* (logical)	true
lnestf	Halo point communication parameter (logical)	true
ldiagt	Diagnostic print parameter (logical)	false
lmpi2	MPI-1/MPI-2 parameter (logical) false = MPI-1 true = MPI-2	false

2.2.1.2 Data paths and data flow

Understanding the control of input and output data is the key to understanding the logic and flow of the atmospheric analysis routine. Figure 2 illustrates the input/output (I/O) structure of *coama.F*.

Each input must be specified in *dsetnl* to properly configure the COAMPS simulation for an idealized or real data assimilation simulation (Table 2.3).

The *dsetnl* parameters indicate the location and type of data required to establish the initial conditions. Before the analysis can be run, the user must obtain and specify the directory locations of these initial input data. Either idealized or real data can be used to provide the initial fields. Furthermore, real-data model runs are performed in either an operational or research and development (R&D) mode.

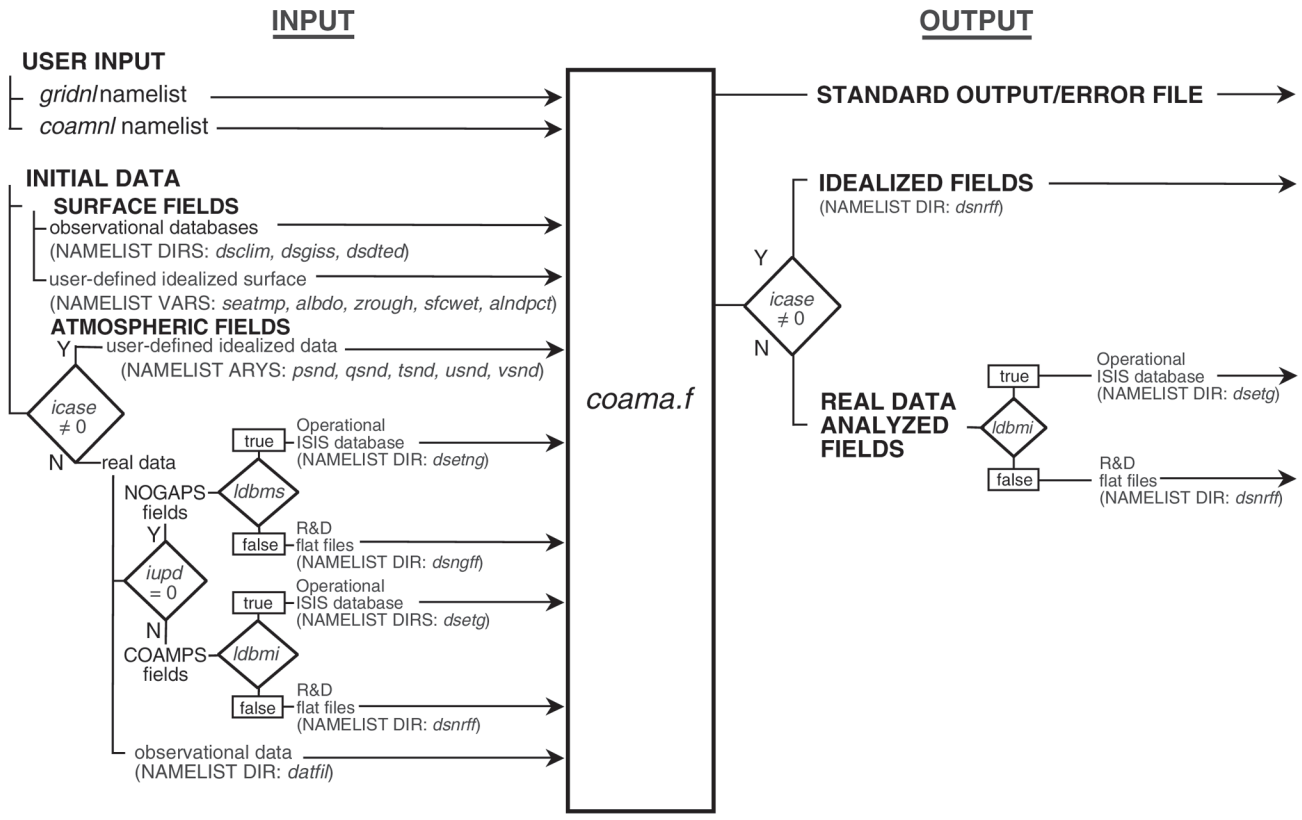


Figure 2
Flow diagram of input/output for the analysis routine *coama.f*. The input requirements for the analysis routine include user input from the namelists, as well as initial surface and atmospheric fields. The analysis routine outputs a standard output/error file and the idealized or analyzed real data fields for the forecast program. For reference, corresponding namelist parameters are shown in parentheses. (From Haack 1996)

Table 2.3 — *dsetnl* Data Path Namelist Variables

Type of Model Run (Namelist variables)	Directory Locations of Initial Input Data (Namelist variables)
operational or research and development (*ldbms, ldbmi, ldbmo*)	surface databases (*dsclim, dsgiss, dsdted, dsland, dslanu*)
idealized or real data (*icase*)	observational data (*datfil*)
data assimilation update (*iupd, loi, loimf*)	NOGAPS fields (*dsetng, dsngff*) COAMPS fields (*dsetg, dsnrff*)

Operational use of COAMPS is accomplished by setting the three database logical variables (*ldbms, ldbmi, ldbmo*) to true. The operational COAMPS will access the Fleet Numerical Meteorology and Oceanography Center's (FNMOC) operational databases, which are managed by the Integrated Stored Information System (ISIS). For R&D purposes, COAMPS reads in real-data initial input fields from flat files using 32-bit IEEE unformatted Fortran I/O. In this case, the user must obtain the required data (first-guess fields, surface databases, observations, and lateral boundary conditions) in the required format (32-bit IEEE unformatted Fortran files), and specify the directory location in which each resides (*dsclim, dsgiss, dsdted, dsland, dslanu datfil, dsetng, dsngff, dsetg, dsnrff*).

After the initial input fields are processed in *coama.F*, they are written out to either: (1) restart files for idealized R&D output, (2) the ISIS database for operational output, or (3) flat files for real-data R&D output.

2.2.2 Setup horizontal grid

After the input parameters have been specified in the *coamnl* namelist, the computational grid for COAMPS must be set up. This is accomplished through routines *nstlvls.F*, *chekmn.F*, *checkz.F*, *bcindx.F*, *grid.F*, *hm2uv.F*, and *xyn.F*, which specify the model domain space, boundary condition indices, map factors, and grid nesting. These routines all contain error checks to ensure that the grids are configured in an acceptable manner. Table 2.4 lists namelist variables that define the parameters for COAMPS horizontal grids.

Table 2.4 — COAMPS *gridnl* Namelist

Name	Type	Description	Default Values
alnnt	real	standard longitude of grid	240
delx	real	grid spacing in x-direction	81000
dely	real	grid spacing in y-direction	81000
ii	integer	coarser mesh x grid point	1 (for all grids)
iref	integer	i-coordinate of reference point	1 (for all grids)
jj	integer	coarser mesh y grid point	1 (for all grids)
jref	integer	j-coordinate of reference point	1 (for all grids)
kka	integer	number atm vertical levels (layers)	30
kko	integer	number ocn vertical levels (layers)	1
lnmove	logical	(true) moving nest indicator	false (for all grids)
m	integer	number x grid positions	61
n	integer	number y grid positions	61
nnest	integer	number nested grids	1
npgrid	integer	parent grid number	1,1,2,3,4,5,6
nproj	integer	grid projection number	2
phnt1	real	1st standard latitude of grid	60
phnt2	real	2nd standard latitude of grid	30
rlat	real	grid reference latitude	42.5
rlon	real	grid reference longitude	16.5

2.2.3 Setup surface fields

For a model simulation, the user may specify whether the simulation will use real or idealized data with namelist parameter *icase* (=0 real data, >0 idealized data).

2.2.3.1 Idealized data

If idealized data have been selected, the *user_sfc.F* subroutine provides the capability to customize the surface parameters with idealized values through namelist definitions. For example, user-defined, horizontally homogeneous values for the land/sea table, surface temperature, roughness, albedo, and ground wetness arrays are given by namelist parameters *alndpct*, *seatmp*, *zrough*, *albdo*, and *sfcwe* when the namelist flags *ilndflg*, *iseaflg*, *iz0flg*, *ialbflg*, and *igwtflg* are set to zero. When the namelist flag *itopoflg* is set to zero, terrain height is assigned to zero everywhere for all nests.

Idealized initial conditions are typically provided by user-defined profiles of wind, temperature, pressure, and moisture and are assumed to be horizontally homogeneous across the model domain. The profiles are defined by namelist arrays *usnd*, *vsnd*, *tsnd*, *psnd*, and *qsnd*, which are used when the namelist parameter *icase* is non-zero. The *refsnd.F* subroutine converts the initial user-defined reference sounding into MKS (meter, kilogram, second) units. Then, a homogeneous idealized mean state is determined from the data from the *istate.F* subroutine for each nest of the model. Perturbations can be added to the mean state by specifying a value for the namelist

parameter *icase*. The value of *icase* is then used in a subroutine *flds[icase_value].F*, which allows the user to assign field perturbations.

In the case of nonhomogeneous idealized fields, set the namelist parameter *jcm2fg*=1 to interpolate the outer coarse mesh fields to the inner nests. The interpolation is performed in the *icm2fg.F* subroutine. The idealized surface initial conditions are created by the analysis. The idealized conditions are initialized at the beginning of the forecast by calling the idealized driver subroutine *ideal_initl.F*.

2.2.3.2 Real data

If a real-data simulation has been selected, the atmospheric surface characteristics for each nest are determined by routines *sfcpar.F*, *getsst.F*, and *soiltp.F*. (Note: the ocean analysis fields (such as sea surface temperature and sea ice fields) are determined from COAMPS Ocean Data Assimilation (CODA)). In the *sfcpar.F* routine, the surface fields for land/water mask (*rland*), albedo (*albedo*), ground wetness (*gwet*), surface roughness (*z0*), and sea surface temperature (*csst*) are obtained from a global climatology database. The land/water mask is obtained from a high-resolution (400-m) database using data path *dsland*. It is recommended that background albedo, ground wetness, and roughness be obtained from high-resolution databases maintained by the U.S. Geological Survey (USGS). The *coamnl* namelist variables *ilandu*=1 (for each nest) and *dslanu* for the correct data path assign the locations for acquiring the data. If the data are not available, albedo and roughness may be obtained from the Goddard Institute for Space Studies (GISS) to replace the climatological values.

The model surface topography is generated in subroutine *sfcpar.F* from either a 20-km or 1-km terrain dataset. There are two options for determining model topography: envelope or silhouette, controlled by *coamnl* parameter *lsilhout* (false=envelope, true=silhouette). The envelope method computes the mean terrain from the database, and the user can add any fraction of standard deviations of the terrain to the mean through namelist variable *sdmult*. The standard deviation file is computed from the 20-km terrain dataset.

When implementing the silhouette method, determination of terrain is controlled by parameters *iwvlng*, *nsrch*, *silwgt*, *silmax*, and *nslflt*. The method uses a 2**delx* or 4**delx* grid based on *iwvlng* (2 is preferred). The size of the box placed around a given grid point in terms of the number of topography grids in the x- and y-direction is determined by *nsrch*. This parameter is currently set to half of the grid resolution. The terrain weighting is determined by *silwgt* (a value between 0 and 1). A value of *silwgt*=1 fully weights the silhouette average (average of the highest peaks in the x- and y-directions). For *silwgt* values less than one and greater than zero, the topography is estimated as the average height of all the points multiplied by *1.0-silwgt*. A value of *silwgt*=0 uses an average topography value within the sample box.

If *nslflt*=1, the topography of the silhouette field will be filtered. It is recommended that topography be created by using the 1-km terrain dataset and silhouette method with default silhouette parameters. After the surface fields have been determined, the topography is blended across the mesh boundaries in *tmatch.F* and topography gradient arrays are computed in *atopo.F*.

The resolution of the sea surface temperature is obtained from either the 1° NOGAPS field, a 125 × 125-km hemispheric grid derived from the Optimum Thermal Interpolation System (OTIS), or from COAMPS history field in the *getsst.F* subroutine. If those sources are not available, the climatological value is used. The sea surface temperature is the default value when CODA is not used.

In subroutine *soiltp.F*, values for the deep soil temperature array are generated from climatological data.

2.2.4 Read and process input data

When working with real-data simulations, COAMPS must read and process input data from either NOGAPS or COAMPS. These initial input fields, referred to as first-guess fields, can also be updated with observations using the Multivariate Optimum Interpolation (MVOI) scheme. The first guess can be obtained from either a NOGAPS

analysis or forecast, or from a previous COAMPS forecast valid at the desired date-time-group (history fields) as shown in Figure 3.

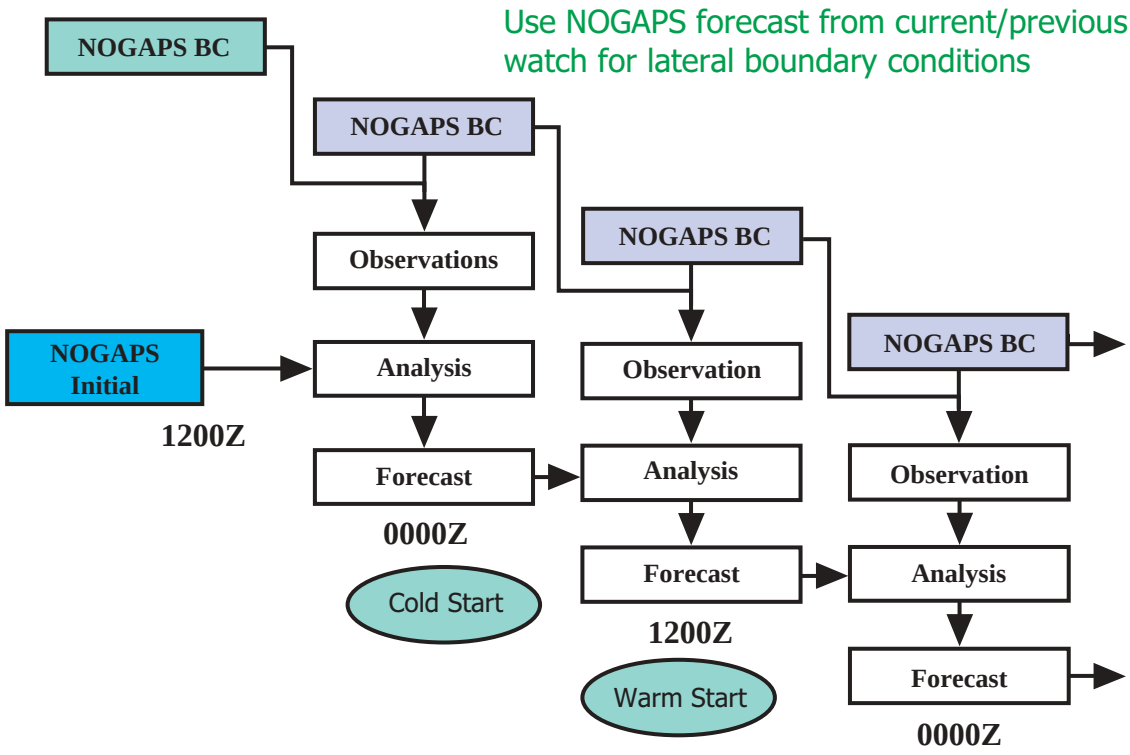


Figure 3
Schematic illustration of COAMPS data assimilation cycling. This example has an update cycle of 12 hours.

2.2.4.1 Specify mean state

The basic state profiles at both mass and w-levels are specified through subroutine *astate.F*.

2.2.4.2 Update surface fields

Routines *gethty.F* and *nethty.F* update the two-dimensional arrays for surface parameters of each nest. The namelist parameter *lnrhty* controls the update of fields for ground wetness and 10-m winds. If *lnrhty*=false, then ground wetness and 10-m winds are not updated with COAMPS first-guess fields. The fields for ground wetness, surface roughness, and surface temperature that were initially obtained from the global climatology database are assigned the values of the COAMPS history fields. Other surface fields that are updated include the 10-m winds, boundary layer depth, surface sensible and latent heat fluxes, and wind stress. If history fields are not found, this latter set of fields is initially set to zero.

2.2.4.3 Process first-guess fields

The procedure for obtaining the first-guess fields begins with routine *getng.F*. The NOGAPS fields for sea level pressure, *u*- and *v*-component winds, geopotential height, temperature (if available), vapor pressure (below 300 hPa), and dewpoint depression (if requested by parameter *lqanl*=true) are obtained.

These fields are specified at 1° horizontal resolution and at pressure levels defined by namelist parameter *pr* (with *lm* levels). The fields are interpolated to the coarse mesh grid points in subroutine *ng2fg.F*. Subroutine *fg2fg.F* interpolates the coarse mesh to the inner nests. If a previous COAMPS forecast is available and the namelist flag *iupd*=2, then subroutine *getfg1.F* is called to overwrite the first-guess fields with COAMPS history

fields for each nest. Both routines *getng.F* and *getfg1.F* search backward through seven date-time-groups in an attempt to provide COAMPS with the most recent data. Table 2.5 lists the surface and upper air first-guess fields used by the COAMPS analysis.

Table 2.5 — COAMPS *Surface and Upper Air First-Guess Fields*

Surface Fields	Upper Level Fields
Ground wetness	U and v winds
Ground temperature	Geopotential height
Snow depth	Air temperature
10-meter wind	Vapor pressure
Boundary layer height	Dewpoint depression
Latent heat flux	
Sensible heat flux	
Wind stress	
Sea level pressure	

2.2.4.4 Multivariate optimum interpolation (MVOI)

The COAMPS analysis is based on the multivariate optimum interpolation (MVOI) analysis scheme described in Goerss and Phoebus (1992) and Barker (1992). The MVOI technique uses observational data to compute increments for the first-guess fields (typically a COAMPS forecast). If computed over a statistically significant number of cases, it minimizes the mean squared error of the analysis. The analysis variables for the MVOI are geopotential height, and the *u* and *v* wind components. Finally, the first-guess fields are adjusted based on observational data via a MVOI analysis performed in the subroutine *oi_anl.F* when namelist *loi*=true.

The *oi_anl.F* subroutine determines height and momentum increments based on differences between the first-guess fields in the coarse mesh and observations (i.e., rawinsonde, SSMI, satellite-derived data, etc.). These increments are added to the first guess fields and smoothing is performed in subroutine *anlfld.F*. The same procedure can be used to obtain MVOI increments directly on the inner nest first-guess fields (*loimf*=true). Alternatively (*loimf*=false), the coarse mesh MVOI increments can be interpolated to the inner nest grids and then added to the inner nest first-guess fields using subroutine *netinc.F*. However, this procedure is less effective at maintaining the mesoscale structure of the observations.

Using a Cressman scheme, additional analyses are available on the pressure levels for temperature (*ltanl*=true) and dewpoint depression (*lqanl*=true), as well as for surface temperature (*ltanls*=true). If *loi*=true, then COAMPS will perform the MVOI using volume sizes as determined by the user. The overlapping volume sizes are dimensioned (*ivol*, *jvol*). If namelist parameter *iloi*=0, then *ivol* and *jvol* are the number of COAMPS grid points in the x- and y-directions, respectively. If *iloi*=1, then *ivol* and *jvol* are the x- and y-distance for the sides of the volume in kilometers. Examples 1 and 2 illustrate the recommended method for indicating volume sizes for the COAMPS MVOI.

Example 1: *iloi* = 0, *ivol*=10, *jvol*=15
COAMPS will perform analysis with volume sizes of 10×15 grid points.

Example 2: *iloi* = 1, *ivol*=60, *jvol*=90
COAMPS will fit volume sizes using *oivol.F* to be approximately 60×90 km.

The COAMPS analysis has the capability to include NOGAPS grid point data as pseudo-rawinsondes. If namelist parameter *lpseud*=true, then COAMPS creates pseudo-observations for the MVOI. Typically, the MVOI analy-

sis scheme will expand the region in which it searches for available data for use in its analysis volume. The maximum number of expansions is controlled by namelist parameter *noiexp*. Search regions increase by a factor of three for each expansion, with the initial search region three times the size of the analysis volume. The size of the analysis volume for pseudo-observations is controlled by namelist parameters *ibog1* and *jbog1*. Namelist parameter *iraob1* determines the threshold number of rawinsondes in each volume. If the number of rawinsondes in a volume of size (*ibog1* × *jbog1*) is less than *iraob1*, then a pseudo-observation is created at the center of the volume. Example 3 illustrates the recommended method for creating a pseudo-observation.

> Example 3: *lpseud*=true, *ibog1*=7, *jbog1*=11, *iraob1*=2, *noiexp*=3
> COAMPS will create volumes of size 7 × 11 grid points. COAMPS will search for data in a region of 21 × 33 grid points first and expand a maximum number of three times. If the number of rawinsondes found in the search region is less than two, then a pseudo-observation will be created at the center of the volume.

2.2.4.5 Boundary condition options

The atmospheric analysis prepares the lateral boundary conditions for the forecast model. The boundary options are determined by *gridnl* namelist parameter *ianal*. If *ianal*=0, then only boundary condition tendencies are computed and output for only the coarse domain. If *ianal*=1 (default), then the analysis is performed and boundary tendencies are computed. If *ianal*=2, then only the analysis is performed. If *ianal*=3, then only Nowcast options are used (e.g., only the analysis is performed). When *ianal*=3, additional ocards (a list of output fields) are created to output model forecast fields at the Nowcast times (controlled by the namelist variable *ncast_start*, *ncast_end*, and *ncast_cycl*) to create the first-guess fields for the subsequent analysis during the next update cycle.

The boundary conditions for the prognostic fields must be supplied to the coarse mesh at a given time interval specified by namelist parameter *itauin*. For a specified forecast length (*itauf*), the boundary values are extracted from the NOGAPS forecast fields in subroutine *getbdy.F* to the COAMPS sigma surfaces. The tendencies (perturbation Exner) function, water vapor mixing ratio, potential temperature, *u* velocity, *v* velocity, and *w* velocity) are computed and written to a file in subroutine *tendbd.F*. If the NOGAPS forecast is missing at the boundary condition output time, the analysis program will attempt to use the NOGAPS forecast at a later time to compute the tendencies. If the two available NOGAPS forecast times are more than 12 hours apart, the analysis will not write any boundary files. Consequently, the forecast model will stop at that time due to missing boundary files. The number of NOGAPS pressure levels used to compute the boundary tendencies can be specified using namelist variables, *lmbc* (namelist *gridnl*) and *prbc* (namelist *coamnl*).

2.2.4.5.1 Data assimilation setup

An important component of the COAMPS analysis is data assimilation. Four important namelist parameters control the boundary condition fields output by the COAMPS analysis for data assimilation: *icycle* (in hours), *itauf*, *itauin*, and *kgetbc* (all in hours, minutes, seconds). Parameter *icycle* determines the time interval in which data assimilation is performed. Parameter *itauf* determines the ending time in which NOGAPS fields will be used to compute boundary conditions. Parameter *itauin* determines the time interval to compute boundary conditions from NOGAPS. Parameter *kgetbc* determines the time interval in which the obtained boundary conditions are used in the model forecast (*kgetbc*>=*itauin*). Example 4 illustrates the recommended method for determining COAMPS boundary conditions:

> Example 4: *icycle*=12; *itauf*=48,0,0; *itauin*=6,0,0; *kgetbc*=12,0,0
> COAMPS will run data assimilation every 12 hours, obtain boundary conditions every 6 hours up to 48 hours from the initial time, but only use the boundary conditions every 12 hours.

Three options are currently available for updating the model first guess to customize the COAMPS analysis in a data assimilation mode. These options are controlled by the namelist parameter *iupd*. If *iupd*=0, then COAMPS performs a cold start in which NOGAPS fields are interpolated to the COAMPS grid. The data are used for the first-guess and analysis increments computed on pressure levels and added to the first guess pressure level fields. The pressure level fields are then vertically interpolated to the COAMPS sigma levels and the analysis is complete.

If *iupd*=1, then COAMPS performs a full update in which a previous COAMPS forecast is used as the first guess instead of NOGAPS. If *iupd*=2, then COAMPS performs an incremental update in which a previous COAMPS forecast is used as the first-guess — similar to the full update. The pressure level increments from the MVOI are interpolated first to COAMPS sigma levels and added directly to COAMPS sigma level first-guess fields. The incremental update option is the preferred option because it retains the vertical structure of the COAMPS first-guess field. Note that if *iupd*=1 or *iupd*=2 and COAMPS first-guess fields are not found, then the COAMPS analysis scheme will automatically revert to a cold start.

2.2.5 Output

The output for the atmospheric analysis is performed by three output routines. Subroutines *outhty.F* and *iosfc.F* (for idealized data) write the surface fields while subroutine *outanl.F* writes the analyzed fields at the pressure levels (specified by namelist variable *pr*). The fields written by these subroutines are listed in Table 2.5. If *loi*=true, subroutine *outinc.F* writes the MVOI increments (geopotential height, *u*, *v*, air temperature, and dewpoint depression for *lqanal*=true) for use in data assimilation. Since the basic state arrays are easily recomputed, they are not written by the analysis routine for real-data cases. The files created by the analysis are named following the standard convention based on namelist parameter *idbms*.

2.2.6 Moving and delayed nest setup

Options exist within COAMPS to move and delay the start of nests. The analysis must consider these options because a common terrain and land/water mask must be computed when either is enabled. The option to move nests is controlled by *gridnl* parameter *lnmove*, and the option to delay nests is controlled by *gridnl* parameter *idelay*.

If *idelay* (hours, minutes, seconds) is greater than zero for any nest, then the start of that nest will be delayed until time *idelay*. (Note: nest 1 cannot be delayed, and delay times for children nests cannot be less than parent nests). A check is performed on the number of nests that are delayed to determine the treatment of the topography and land-water mask. If only one nest is delayed (innermost) and there are no moved nests, then the topography and land-water masks are determined at that nest by inserting the fields down from the parent nest to one level up. However, if more than one nest is delayed or if any nests will be moved during the model integration (as determined by *lnmove*=true), then a blended topography and land-water mask must be used to ensure consistency in horizontal and vertical interpolation. The blended fields are computed using the parent grid domain at the child grid resolution (Figure 4).

Care must be taken when establishing which nests are moved and delayed because the associated data arrays can become very large and memory-intensive. The goal is to use the highest resolution possible for surface fields, at the same time realizing that vertical and horizontal interpolations from the parent to child nest are necessary in a delayed or moved nest region.

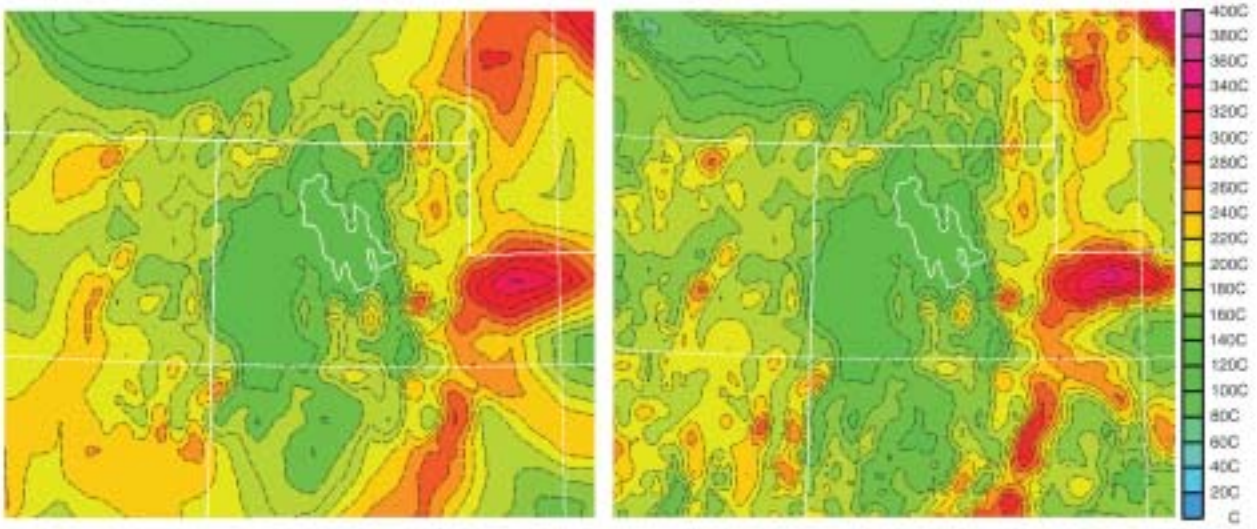


Figure 4
COAMPS topography plots for a 9-km nest over Utah area for move=f (left panel) and move=t (right panel) options.The biggest differences are in the nested domain boundary blend zone (usually 7 grid points) where the move=t option does not have the blend problems of the move=f option because the values are matched at every grid point from the "master terrain" computed at nest 2 resolution on nest 1 domain.

3. Ocean Data Assimilation

3.1 Introduction

This chapter describes the ocean analysis component of COAMPS — the Coupled Ocean Data Assimilation (CODA). CODA is tightly integrated with the COAMPS software design. As such, it fully supports the nested, movable grid structures of COAMPS. CODA can be used in a variety of ways within COAMPS. CODA can be executed in a two-dimensional mode that provides updated sea surface temperature and sea ice concentration lower boundary conditions for the COAMPS atmospheric forecast model. In addition, CODA can be run in a stand-alone mode to provide fully three-dimensional analyses of ocean temperature, salinity, and currents. Alternatively, CODA can be cycled with the ocean forecast model component of COAMPS in a sequential incremental update cycle, providing updated initial conditions for the next ocean model forecast run.

The analysis background or first-guess fields are generated from a short-term ocean model forecast or from a previous analysis. CODA computes corrections to the first-guess fields using all of the observations that have become available since the last analysis was made. The forecast model with the new initial conditions is then run forward in time to produce the forecast. The time interval between successive analyses is called the update cycle and is a user-defined parameter in CODA.

3.2 Method

The method used in CODA is an oceanographic version of the multivariate optimum interpolation (MVOI) technique widely used in operational atmospheric forecasting systems. A complete derivation and description of the MVOI method is provided in Daley (1991). Application to atmospheric systems is given in Lorenc (1981), Hollingsworth and Lonnberg (1986), Lonnberg and Hollingsworth (1986), and Goerss and Phoebus (1992).

The ocean analysis variables in CODA are temperature, salinity, geopotential (dynamic height), and velocity. Geopotential is not directly observed in the ocean, but it is calculated from observations of temperature and salinity by using standard oceanographic methods. In a CODA analysis, all ocean variables are analyzed simultaneously in three dimensions. The horizontal correlations are multivariate in geopotential and velocity, thereby permitting adjustments to the mass fields to be correlated with adjustments to the flow fields. The velocity adjustments (or increments) are in geostrophic balance with the geopotential increments that, in turn, are in hydrostatic agreement with the temperature and salinity increments.

The MVOI problem is formulated in CODA as

$$x_a = x_b + P_b H^T (H P_b H^T + R)^{-1} [y - H(x_b)], \tag{3.1}$$

where x_a is the analysis, x_b is the background, P_b is the background error covariance, H is the forward operator, R is the observation error covariance, and y is the observation vector. The observation vector contains all of the synoptic temperature, salinity, and velocity observations that are within the geographic and time domains of the COAMPS forecast model grid and update cycle.

A forward model is a method of converting a forecast grid variable to an observed variable. The MVOI does not explicitly include any forward models and can only analyze observations that are of the same variable as the model forecast grid. The forward operator in CODA is simply a spatial interpolation of the forecast model grid to the observation location performed in three dimensions. Thus, HP_bH^T is approximated directly by the background error covariance between observation locations, and P_bH^T directly by the error covariance between obser-

vation and grid locations. For the purposes of discussion, the quantity $[y - H(x_b)]$ is referred to as the innovation vector, $[y - H(x_a)]$ is the residual vector, and $x_a - x_b$ is the increment (or correction) vector.

3.3 Error Covariances

Specification of the background and observation error covariances in the analysis is very important. The background error covariances are separated into a background error variance and a correlation. The correlation C_b is further separated into a horizontal C_h and a vertical C_v component. All correlations are modeled as second-order autoregressive (SOAR) functions of the form

$$C_h = (1 + s_h)\exp(-s_h), \tag{3.2}$$

$$C_v = (1 + s_v)\exp(-s_v), \tag{3.3}$$

$$C_b = C_h * C_v, \tag{3.4}$$

where s_h and s_v are the horizontal and vertical distances between two locations (observations or observation and a grid point). The distances are normalized by the geometric mean of the horizontal and the vertical correlation length scales are prescribed a priori at the two locations. CODA allows the horizontal correlation length scales to vary with location and the vertical correlation length scales to vary with depth.

The default horizontal length scales are specified as the first baroclinic Rossby radius of deformation computed from the historical profile archive (Chelton et al., 1998). The Rossby length scales vary from ~10 km at the poles to greater than 200 km in the tropics (Figure 5). Note, however, that the default Rossby length scales can be modified by a user-defined constant of proportionality. Furthermore, the user can specify an input file to CODA that contains the horizontal correlation length scales, thereby bypassing the default specifications based on Rossby radius of deformation scales. This last option may be useful in shallow-water applications where the horizontal correlation length scales are assumed to vary with bottom depth or some other user-definable quantity.

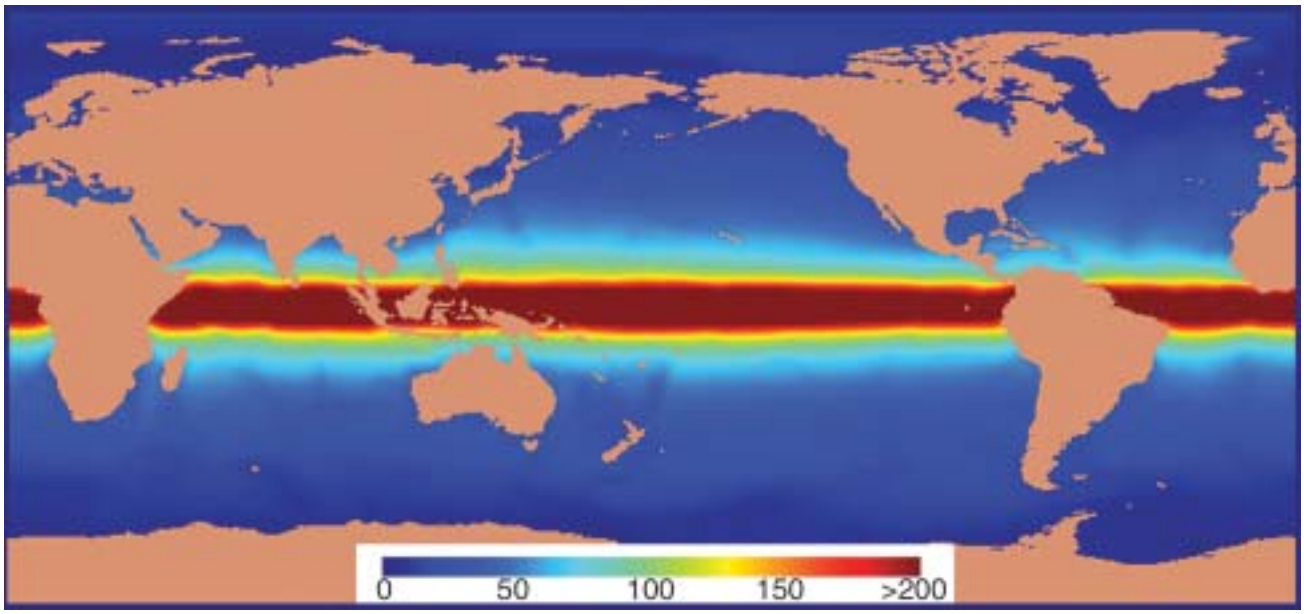


Figure 5
Geographic variability of first baroclinic Rossby radius of deformation computed from the historical profile archive. Rossby length scales are used as the default horizontal correlation length scales in CODA. The Rossby scales vary from ~10 km at the poles to greater than 200 km in the tropics. (From Chelton et al. 1998)

CODA contains a variety of options that can be used to specify the vertical correlation length scales. Vertical correlation length scales can: (1) be constant (or zero), (2) monotonically increase or decrease with depth, or (3) vary with background density (or temperature) vertical gradients. In the latter option, a change in density (temperature) stability criterion is used to define a well-mixed layer. The change in density (temperature) is then scaled by the background vertical density (temperature) gradient. The result of this calculation is a correlation length scale that varies with depth and is large (small) when the water column stratification is weak (strong). Vertical correlation length scales are computed each update cycle from the model forecast density (temperature) fields. This process allows the vertical scales to evolve with time to capture changes in mixed layer depth and undulations in the thermocline, for example.

The univariate horizontal correlation functions shown above are used in the analysis of temperature, salinity, and geopotential, which assumes that all of these variables have the same background error correlations. The formulation of the multivariate background error correlations, however, is derived from the first and second derivatives of the SOAR model function. This form requires calculating the angles between the two locations and specifying a parameter *v*, which measures the divergence permitted in the velocity correlations, and a parameter *u*, which specifies the strength of the geostrophic coupling of the velocity/geopotential correlations. Typically, *v* is set to a constant small value that does not vary with location. This setting will produce velocity increments that are weakly divergent, and assumes that the divergence is not correlated with changes in the mass field. The geostrophic coupling parameter *u*, however, does vary with location. It is scaled to zero at the equator where geostrophy is not defined and in shallow water where the friction terms of the equations of motion dominate the flow.

As mentioned previously, a full derivation of the multivariate horizontal correlations is provided in Daley (1991, chapter 5). Using the derivatives of the geopotential SOAR correlation model and converting from natural to rectangular coordinates, the correlation functions of the possible combinations of geopotential and velocity are shown in Figure 6. There are nine possible combinations, but only one form of the cross-correlations between geopotential and the velocity components is shown for clarity. In Figure 6, the geostrophic coupling parameter *u* is set to 1, and the divergence is permitted to be nonzero ($v = 0.1$).

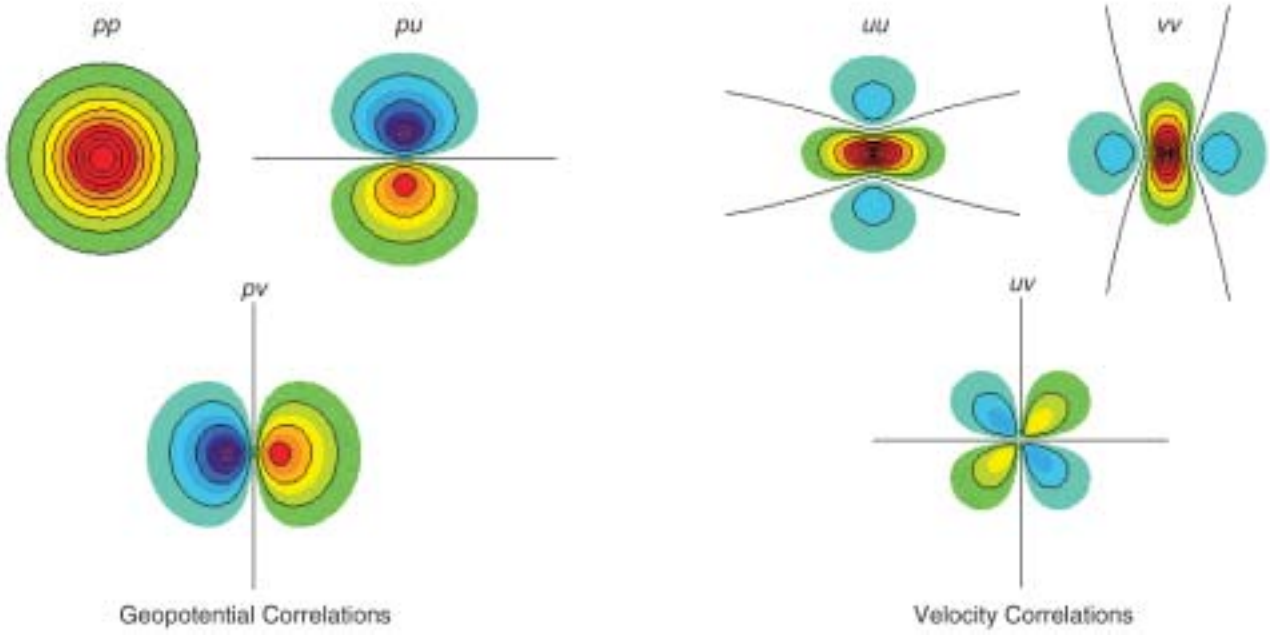


Figure 6
Auto and cross correlations of geopotential (*p*) and velocity components (*u*,*v*) computed from multivariate correlation functions. The geopotential correlation is modeled as a SOAR function. Cool colors indicate negative correlations, and warm colors indicate positive correlations.

The background error variances in CODA (E_b^2) are permitted to vary with location, depth, and analysis variables. Background error variances are poorly known in the ocean. They are likely to be strongly dependent on model resolution and other factors such as atmospheric forcing errors and ocean model parameterization errors. Accordingly, the error variances in CODA evolve with time and are computed prior to an analysis from a time history of the innovation vectors or analyzed increment fields.

Forecast error variances computed from the innovation vectors are known only at the observation locations, and thus can only be stratified by depth and analysis variable in CODA. However, forecast error variances computed from the analyzed increment fields are known at the model grid points and, in addition to depth and analysis variable, the error variances can change with location. This option assumes that the background error covariance structure is reasonably well defined in the statistical interpolation of the observation innovations to the grid locations in CODA.

Error variances computed from the analyzed increment fields include a climate error growth rate parameterization to account for the inherent sampling limitations of the ocean observing systems. In the long-term absence of ocean observations, the background error variances are slowly restored to climate variability values by using a climate decorrelation time scale that is defined independently for each analysis variable. In practice, the background error variances, computed via either method, evolve to a steady-state condition over time and reflect the long-term average prediction error variances of the model forecast at the analysis update time. In a cold start scenario, when the analysis is initialized from climatology, the background error variances are specified to the ocean climate variability interpolated in time and space to the model start time and grid locations. The climate variability is then slowly replaced by the analyzed increment fields in the computation of the error variances, provided adequate sampling of the ocean observations in time and space.

The observation errors and the background errors are assumed to be uncorrelated, and errors associated with observations made at different locations and at different times are also assumed to be uncorrelated. As a result of these assumptions, the observation error covariance matrix R is set equal to $1 + e_o^2$ along the diagonal and zero elsewhere. Note that e_o^2 represents observation error variances that have been normalized by the background error variances interpolated to the observation location ($e_o^2 = E_o^2 / E_b^2$**)**. Observation errors are computed as the sum of a measurement error and a representation error. Measurement error variances are specified as input parameters in CODA with one exception.

Measurement errors reflect the accuracy of the instruments and the ambient conditions in which the instruments operate. These sources of error are fairly well known in the ocean, although the magnitude of the measurement errors can change in time due to calibration drift of the instruments and other factors. Representation errors, however, are a function of the resolution of the model and the resolution of the observing network, and are much more difficult to quantify. For example, the model state represents an average around a grid point. The resolution of the model grid may not resolve ocean eddies or internal oscillations that, however, are measured by the observations. Thus, representation error is a measure of the error caused by the scales of variability being smaller than what can be resolved by the model and observed by the observation network.

Representation error is estimated in CODA by using a simple technique. Each observing system is assigned a characteristic sampling scale. The sampling scale is normalized by the model grid resolution, and the normalized values are used to weight an assigned representation error. For point measurements, the sampling scale is set to 1 km, but for satellite systems, which sample an area on the surface of the ocean, the sampling scale can vary. The spatial averaging properties of the remotely sensed observations tend to filter small-scale structure and reduce representation error compared to point measurements. Increasing model grid resolution is the only way to reduce the representation error of point measurements.

The consistency of the specified error variances with the innovation vector is estimated by the a posteriori χ^2 diagnostic (Bennett 1992). When normalized by the number of observations, the expected value of the χ^2 diag-

nostic is one. χ^2 values greater than one indicate that either the background and/or the observation error variances are specified as too small or erroneous observations are being assimilated. χ^2 values less than one indicate that either the background or the observation error variances are specified as too large. CODA routinely computes χ^2 diagnostics for all observing systems and analysis variables at each update cycle. The computed χ^2 values indicate whether the observation and/or the background error variances need to be modified. A user-defined scalar value is used in CODA to modify the error variances accordingly. Monitoring the χ^2 diagnostic in subsequent executions of CODA will determine if the error variance scaling produces an appropriate response. Note, however, that inferences made from the χ^2 diagnostic require very large sample sizes from many time integrations of the assimilation system.

The exception to the offline specification of the observation error variances is that of the geopotential observations. Recall that geopotential is not directly observed in the ocean. Instead, it is computed from the temperature and salinity observations during a CODA analysis. Geopotential is computed by integrating the specific volume anomaly from a level of no motion to the surface. The specific volume anomaly α is computed from the temperature and salinity observations using the UNESCO equation of state for seawater (Fofonoff and Millard, 1983). The errors in specific volume anomaly E_a^2 are computed from the errors in the potential temperature E_q^2 and salinity E_S^2 observations using the partial derivatives of the equation of state with respect to the potential temperature and salinity according to

$$E_\alpha^2 = (\delta\alpha / \delta\theta) * E_\theta^2 + (\delta\alpha / \delta S) * E_s^2 . \tag{3.5}$$

The specific volume anomaly errors are then integrated through the water column from the user-defined level of no motion to the surface in the same way as the geopotential itself is integrated. Geopotential errors at and below the level of no motion are set to a small value.

It is clear that specification of the background error covariances in CODA can be improved. First, the geostrophic coupling parameterization is most likely scale dependent and invalid at very small (nonhydrostatic) scales in the ocean. In addition, divergent flow is likely to be correlated with changes in the mass fields during periods of intense atmospheric forcing such as tropical cyclones. Second, it is very likely that horizontal correlation length scales vary in the vertical and that vertical correlation length scales vary in the horizontal in the ocean. These nonseparable error correlation models have not yet been incorporated in CODA.

More importantly, though, is the need for flow-dependent forecast error correlations. It is intuitive that error correlations across an ocean front are characteristically smaller than error correlations along the front. This sort of information can be made part of the CODA MVOI analysis by a simple transformation of the vertical coordinate system from pressure to isopycnal space (Riishojgaard, 1998). Currently, vertical correlation models use the distance between the depths of any two observations. If this distance is computed on the basis of observed differences in density, then the error correlations are considerably more flow-dependent. The innovations are spread along rather than across isopycnal layers, which do not necessarily co-vary with pressure surfaces in the ocean. However, the transformation from pressure to isopycnal coordinates must use forecast model density fields and, thus, depends strongly on the accuracy of the model forecast. The transformation is essentially equivalent to the assumption of a perfect model and may not prove to be very useful if the background density field is inaccurate.

3.4 Ocean Observations

CODA makes full use of all sources of the operational ocean observations. Table 3.1 lists the ocean observing systems currently assimilated along with typical global data counts per day for each of the systems. From Table 3.1, it is apparent that most ocean observations are remotely sensed and measure ocean variables only at the surface.

Table 3.1 — Ocean Observation Data Sources — NRL Coupled Systems

Data Source	Specification	Estimated Number of Daily Observations
AVHRR GAC Satellite SST	NOAA 16 8-km resolution (day, night, relaxed day retrievals)	300,000
AVHRR LAC Satellite SST	NOAA 16 2-km resolution (day and night retrievals)	1,500,000
GOES Satellite SST	GOES 8, 10 12-km resolution (day and night retrievals)	5,000,000
In Situ SST/SSS	Surface ship, fixed and drifting buoys, CMAN, TRACKOB	15,000
Subsurface Temperature and Salinity Profiles	- XBTs, CTDs (TESACS), and PALACE floats - fixed buoys (TAO and PIRATA), thermistor chain of drifting buoys	750
Sea Surface Height Anomaly (SSHA)	Altimeter (TOPEX, ERS2, and GFO)	100,000
Sea Ice Concentration	SSM/I (DMSP F13, F14, F15)	1,000,000

The lack of synoptic real-time data at depth severely limits the ability of the data assimilation system to resolve and maintain ocean fronts and mesoscale eddies at correct locations and times. Subsurface properties in the ocean, therefore, must be inferred from surface-only observations. The most important observing system for this purpose is satellite altimetry, which measures changes in sea level. Changes in sea level are strongly correlated with changes in the depth of the thermocline in the ocean, and the ocean dynamics generating sea-level changes are mesoscale eddies and meandering ocean fronts. CODA supports two alternative methods to the assimilation of satellite altimeter observations of sea level anomaly. Note that in both methods the altimeter data are not directly used to change the background fields. Instead, altimeter-derived synthetic observations are combined in the analysis with in situ observations and the model background. Selection of the altimeter assimilation method is via a namelist option.

One approach is the assimilation of synthetic temperature profiles computed using the Modular Ocean Data Assimilation System (MODAS) database. The MODAS database contains regression coefficients derived from the historical profile archive that relate steric height anomalies to climate temperature anomalies at depth. Fox et al. (2002) provides a complete description of MODAS. The MODAS synthetic temperature profiles are appended to the other real-time observations and assimilated in the same way as any other observation but with unique error characteristics specified. The error variances of the MODAS synthetic profiles depend on the accuracy of the sea level predictor field and on the magnitude of the residuals of the historical profile regressions relating steric height and temperature at depth. The residuals of the MODAS regressions vary with location, depth and time of year. MODAS synthetics have marginal skill in some areas of the world's oceans due to sampling limitations of the historical profile data and non-steric signals in the altimeter data. In these areas MODAS tends to return and, therefore, continually restores the model forecasts to the MODAS climate mean conditions.

An alternative approach is the direct assimilation of observed sea level changes using a modified form of the method developed by Copper and Haines (1996). In this approach, the model forecast density field is adjusted to correct errors between the model height field and the height field measured by the altimeter. The adjustments are computed by

$$\Delta p_s + g\int_{z_b}^{z_t} \Delta\rho_z dz = \Delta p_b\,, \tag{3.6}$$

where Δp_s is the surface pressure change measured by the difference between the model background and the satellite altimeter observation $\Delta\rho_z$ is the change in density at level z, g is the gravitational constant, and Δp_b is the bottom pressure change. The depth range over which the density corrections are made is constrained to be between the mixed layer z_t and a level of no motion z_b where Δp_b is assumed to be zero. Output of the integration is in the form of innovations of temperature and salinity from the background field and, as with MODAS synthetic profiles, the innovations are appended to the real-time data stream for the assimilation. The error variances of the innovations are computed as the sum of the respective background error variance plus residual error from the iterative fit of the density adjustments to the observed change in sea level. An advantage to this method is that it relies on model dynamics for a priori information rather than statistics fixed at the start of the assimilation. Furthermore, the method simultaneously computes adjustments to the model temperature and salinity profiles. As a result, it does not introduce spurious water masses into the model. A disadvantage is that the method cannot explicitly correct for errors in the stratification or long-term drift of the water mass characteristics in the model.

While having the potential of adding important information in data-sparse areas, the number of altimeter-derived supplemental observations computed using either method can greatly exceed and overwhelm the in situ observations in the analysis. Accordingly, the synthetic observations are thinned in two steps prior to the analysis. In the first step, it is assumed that directly observed temperature and salinity profiles are a more reliable source of subsurface information wherever such observations exist. Altimeter-derived subsurface profiles, therefore, are not generated within one correlation length scale of an in situ observation. In the second step, the change in sea level measured by the altimeter must exceed a threshold value to trigger the generation of a synthetic observation. The threshold value is defined to be the noise level of satellite altimeters, which is in the range of 2-4 cm. The actual threshold value used in the analysis is a namelist parameter

Finally, all observed temperatures must have a companion salinity observation because CODA is a multivariate method. Salinity is routinely measured in some observing systems, such as PALACE (Profiling Autonomous Lagrangian Circulation Explorers) floats, whereas other systems measure temperature only. Accordingly, for systems that report only temperature, salinity is estimated from the observed temperature using regression relationships derived from the historical profile archive stored in the MODAS database. The observation error variances of the derived salinity values are estimated from the residuals of the MODAS regressions, which vary in both space and time.

3.5 Quality Control

All ocean observations are subject to quality control (QC) procedures prior to assimilation. The primary purpose of the QC system is to identify observations that are obviously in error, as well as the more difficult process of identifying measurements that fall within valid and reasonable ranges but are erroneous. QC procedures common to all data types include land/sea boundary checks, background field checks against climatology, and forecast fields using appropriate background error variances. QC procedures unique to different data types include: location (speed) test for drifting buoy and ship observations, instrumentation error checks for expendable bathythermographs and profiling floats, sensor drift for fixed and drifting buoys, and large-scale bias detection for satellite retrievals of SST. Cross-validation checks are also performed to ensure consistency between variables. Examples of this include SST and sea ice concentration, to check for impossible ice conditions, and wind speed and daytime satellite SST retrievals, to check for biases due to skin temperature effects.

The need for quality control is fundamental to a data assimilation system. Accepting erroneous data can cause an incorrect analysis, while rejecting extreme but valid data can miss important events. Decisions made at the quality control step affect the success or failure of the entire analysis/forecast system. The outcome of the QC processes on an ocean observation is a probability of gross error. The magnitude of an acceptable gross error probability is a user-defined parameter in CODA, and thus is an integral component of the space/time queries performed on the QC data files when gathering observations for assimilation.

A secondary use of the QC system is the creation and maintenance of an analysis-forecast innovation database for use in the a posteriori computation of the parameters required in the MVOI. Statistical analysis of the innovations is the most common, and the most accurate, technique for estimating observation and forecast error covariances, and the method has been successfully applied in practice (Hollingsworth and Lonnberg 1986).

3.6 Pre-processing Ocean Observations

Prior to assimilation, CODA performs a wide range of pre-processing functions on the quality-controlled ocean observation data sets. Many of these functions are under user control via ocean analysis namelist options. For example, combining separate satellite missions into one data type effectively reduces the high-volume satellite data, assuming measurement errors of the satellites are the same. These data can be further reduced by the formation of super-observations. The super-observations are innovations averaged into bins dependent on grid resolution and observation data type. For temperature observations, super-observations are stratified further by water mass using Bayes[1] rule. The Bayesian water mass classification statistics have been computed for the western boundary current regimes of the world's oceans from frequency distribution analyses of satellite SST retrievals. Water mass classification prevents averaging observations across ocean fronts and eddies.

Thinning of observations is a necessary step in the analysis to remove redundancies in the data and minimize horizontal correlations among observations. However, the algorithm used to thin data in CODA is adaptive. As the model grid resolution increases, the actual number of innovations averaged into a super-observation decreases until eventually the original data are directly assimilated. This feature is very useful in a nested COAMPS analysis in which the nested grids telescope down to the desired forecast model grid resolution.

Pre-processing options on the profile data include: (1) specification of selection criteria to ensure adequate sampling in the vertical, (2) vertical extension of the profiles from the last observed depth to the bottom, and (3) assimilation of profile observations at observed levels or after vertical interpolation to model levels.

Timely receipt and quality control processing of observations are important issues for CODA, especially if the system is run in real time. Unfortunately, not all observing systems routinely report in real time. For example, observations from profiling floats and satellite altimeter sea level anomalies are often received days to even weeks after observation time. Requiring observations to be synoptic about the analysis time will effectively exclude many important sources of ocean observations in a real-time application.

CODA has been designed to handle the inevitable delays in the receiving and processing of ocean observations at the production center in several ways. First, CODA saves and maintains a table of the youngest observation times found for all observing systems at the update cycle. These "look-back" times are used in the space/time query of the observation data files when forming the incoming data set for the next analysis. Observation time can be defined in CODA as the true sampling time of the observation or the time the observation is received at the center. By selecting observations on the basis of receipt time, data from observing systems that report late can still be included in the analysis. The observation errors of late-receipt near-surface data are increased but, because of the long time scales in the ocean, observations below the seasonal thermocline are considered synoptic for many weeks. Second, the design of CODA allows it to be successfully run multiple times using the same date-time-group. This type of execution is termed a post-time analysis and is typically performed after the real-time analysis, immediately prior to the next forecast run. The purpose of a post-time analysis is to include late-receipt observations in the analyzed corrections to the forecast model initial conditions. In this mode of operation, observations are usually required to be synoptic about the analysis time, and observation times are defined as the actual sampling time.

3.7 Analysis Control

As previously mentioned, the execution of CODA is controlled via namelist parameter input specified in the COAMPS run script. Two namelists are required by CODA. The first is the ocean analysis namelist (Table 3.2),

which specifies all of the CODA control variables. Execution of a fully three-dimensional analysis of temperature, salinity, and currents is simply done by setting the appropriate *locn3d* grid nest value in the ocean analysis namelist to be true.

Table 3.2 — COAMPS Ocean Analysis *oanl* Namelist

Name	Type	Description	Values
blist	char	blacklisted call signs	no default call signs
case	integer	option for creating initial conditions	0 = real data case (default) 1 = GDEM seasonal climatology 2 = constant density
clm_scl	real	climate decorrelation time scale (hours) for ice, sst, ssh, mvoi, and swh.	default = 480, 720, 720, 720, 96
cold_start	logical	force cold start on nest 1 (true)	default = false
debug	logical	generate diagnostics (true) for 1) pooling of profile moorings and profile rejections (fort.32), 2) profile inflexion point and standard level data and vertical extension results using background fields (fort.33), 3) geopotential profile observations (fort.34), 4) observation and prediction errors (fort.35), 5) listing of MVOI observations (fort.36), 6) synthetics (direct and MODAS) for MODAS including rejections and editing results (fort.31)	default = false (for all)
dbv_opt	char	source bathymetry database	DOD = DBDBV from NAVOCEANO (default) SAS = Smith and Sandwell all = all sources (DBDBV plus Smith and Sandwell)
dbv_res	real	minimum resolution of bathymetry to retrieve from variable resolution database (minutes)	default = 5
del_ssh	real	minimum change in SSHA to force generation of a synthetic profile	default = 0.02
del_sst	real	minimum change in SST to force sampling of analyzed SST field	default = 0.2
den_ds	real	change in density definition for MLD	default = 0.15
deny	integer	data types to deny analysis (see "coda_types.h" for type codes)	default = 0
direct	logical	(true) perform direct assimilation SSHA	default = false
dv_dz	real	vertical gradient length scale (units are dependent upon vc_mdl selection)	default = 0.3
ebkg	real	background error tuning factors for ice, sst, ssh, temperature, salinity, geopotential, velocity, and swh.	default = 1 (for all)
eobs	real	obs error tuning factors (see "coda_types.h" for type codes)	default = 1 (for all)

Table 3.2 (continued) — COAMPS Ocean Analysis *oanl* Namelist

Name	Type	Description	Values
emdl	char	background error option for ice, sst, ssh, temperature, salinity, geopotential, velocity, and swh.	'simple' = homogeneous errors (default for all) 'complx' = non-homogeneous errors
hc_mdl	char	horizontal correlation model options for ice, sst, ssh, mvoi, and swh.	'rsby' = non-homogeneous scales (default for all) 'homo' = homogeneous scales 'locl' = user defined scales
linck	logical	perform innovation error check (true) for ice, sst, ssh , temperature, salinity, geopotential, velocity, direct method synthetics, and swh.	default = false (for all)
linit	logical	(true) perform initialization procedures on cold start fields	default = false
lncom	logical	(true) perform NCOM setup procedures	default = false
locn3d	logical	(true) do 3D analysis on this grid nest	default = false (for all grid nests)
mask_opt	char	grid mask option ("2D" or "3D") (used only when locn3d is true)	default = "2D"
mds_age	real	minimum obs age (hrs) to force synthetic	default = 144
mds_edit	logical	(true) edit MODAS synthetics	default = true
mds_grd	logical	(true) generate MODAS synthetic profile initial conditions on a cold start	default = false
mds_mld	logical	(true) apply modas mld model	default = true
mds_xtnd	logical	(true) extend MODAS synthetics with Levitus climatology	default = true
pool	logical	(true) pool satellite systems for MSP F11, F13, F14, F15; NOAA 14, 15, 16, 17 GAC SSTs; TOPEX, ERS, GFO, JASON, ENVISAT; GOES 8, 10 SSTs; NOAA 16,17 LAC SSTs; Altimeter / Buoy SWH	default = false (for all)
prf_opt	char	profile processing option	obsz = assimilation of profiles on observed levels anlz = assimilation of profiles after interpolation to analysis levels (default)
prf_slct	real	profile selection criteria options for 1) acceptable level probability gross error 2) minimum number of sampling levels 3) minimum ratio of last sampling depth and bottom depth 4) minimum sampling depth (if profile has not sampled water column) 5) maximum acceptable distance between adjacent levels 6) maximum acceptable temperature difference between adjacent levels	default = 0.99, 5, 0.5, 300, 300, 5
prf_time	char	profile time sampling option	obst = select profiles based on time profile was observed (default) rcpt = select profiles based on time profile was received at center

Table 3.2 (continued) — COAMPS Ocean Analysis *oanl* Namelist

Name	Type	Description	Values
prf_xtnd	logical	(true) extend inflexion point profiles to bottom using first guess fields	default = true
qc_err	real	max acceptable probability gross error for 1) DMSP sea ice 2) AVHRR satellite sst 3) GOES satellite sst 4) in situ sst 5) profile temperature (integrated over levels) 6) profile salinity (integrated over levels) 7) altimeter SSHA 8) altimeter/buoy SWH	default = 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 1, 0.99
rscl	real	rossby radius scaling factor for ice, ssh, multivariate, and swh.	default = 3 (for all)
sal_adj	logical	(true) adjust SSS derived from SST obs for density inversions	default = true
sal_std	real	max number std dev to scale modas salinity from climatology	default = 3
spval	real	missing value (must be < -99)	default = -999
ssh_time	char	altimeter ssh processing option	obst = select obs based on time SSHA was observed (default) rcpt = select obs based on time SSHA was received at center cycl = select obs based on full Topex repeat cycle (10 days)
ssh_std	real	max number std dev to scale altimeter ssh from climatology	default = 1
st_chn	logical	(true) generate "thermistor chain" observations to the base of the mixed layer from SST (see also st_grid)	default = false (for all grid nests)
st_grd	logical	(true) generate SST observations for 3D MVOI from analyzed SST grid	default = false (for all grid nests)
st_ntrvl	integer	SST "thermistor chain" vertical sampling interval	default = 1
swh	logical	(true) perform SWH analysis	default = false
swh_time	char	swh observation processing option	obst = select obs based on time swh observed (default) rcpt = select obs based on time swh received at center
tmp_std	real	max number std dev to scale modas temperature from climatology	default = 4
tol_fctr	real	innovation error check tolerance factor for ice, sst, ssh, temperature, salinity, geopotential, velocity, and swh.	default = 4 (for all)
upd_cyc	integer	analysis update cycle (hours)	default = 24
vc_mdl	char	vertical correlation model options	'mixl' = mixed layer (default) 'dens' = density stratification 'temp' = temperature stratification 'mono' = monotonic 'cons' = constant 'none' = no vertical correlation

Table 3.2 (continued) — COAMPS Ocean Analysis *oanl* Namelist

Name	Type	Description	Values
vol_scl	real	minimum number of correlation length scales in an analysis volume for ice, sst, ssh, mvoi, and swh.	default = 4 (for all)
z_lvl	real	analysis vertical grid (meters). if lncom is true, then the vertical grid is computed from ncom model parameters specified in "omnl.h" and z_lvl is ignored	default = 0, 5, 10, 15, 20, 30, 50, 75, 100, 125, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1750, 2000, 2500, 3000, 70*0

The second namelist, listed in Table 2.4, is the grid definition shared by all COAMPS components. By default, CODA performs sea ice concentration and sea surface temperature analyses on all nested grids defined in the grid namelist.

CODA automatically detects the presence of the appropriate forecast fields valid at the update time in the COAMPS ocean restart directory and performs a sequential incremental update analysis if all of the requisite fields are present. However, if any of the forecast fields are absent from the restart directory, then CODA will use the previous analysis as the first-guess fields in the current analysis. This default mode of execution assumes a persistence forecast from the previous analysis.

Figure 7 shows the flow of information from the observations to the quality control, to the analysis, and ultimately to the forecast model. Note the feedback of the forecast model and prediction errors in the quality control of newly received observations. The entire system can be run in an end-to-end mode, or individual components of the system can be executed separately. Flexible implementation of the ocean QC-Analysis-Forecast components is required for COAMPS to be run in a variety of operational settings, including central sites, regional centers, and onboard ships at sea.

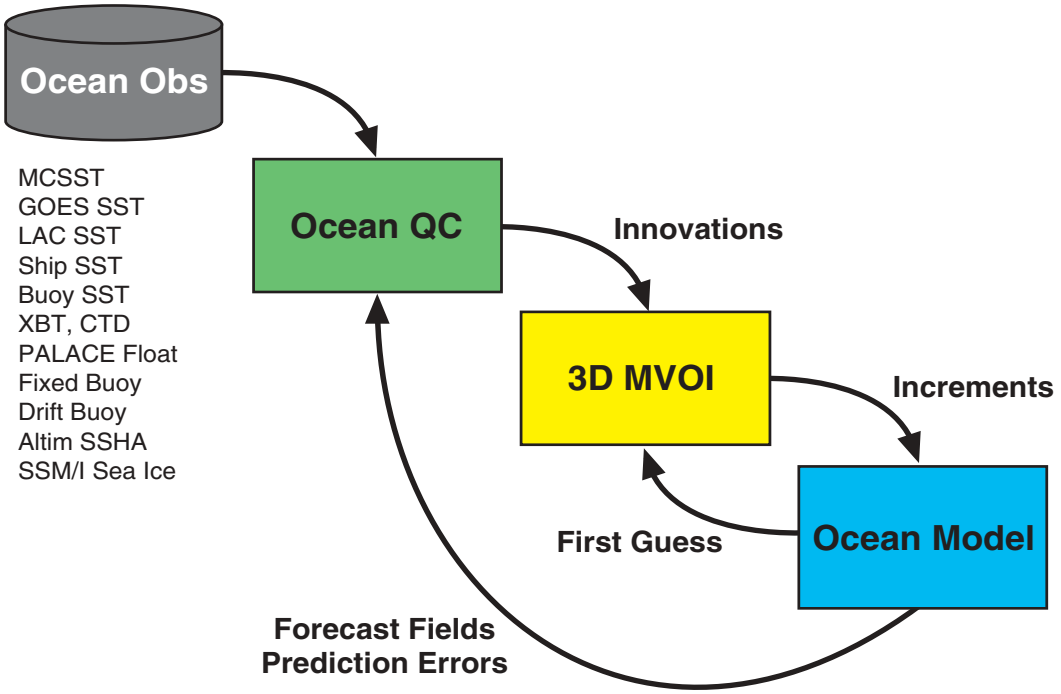


Figure 7.
Ocean analysis and forecast flow diagram

4. Description of the COAMPS Atmospheric Model

4.1 COAMPS Model Numerics

The atmospheric portion of the COAMPS models is comprised of the nonhydrostatic, fully compressible equations of motion following Klemp and Wilhelmson (1978). The adiabatic equations are developed using the equation of state:

$$p = \rho R_D T_v, \tag{4.1}$$

where p is the pressure, ρ is the density, R_d is the dry gas constant, the virtual temperature, T_v,

$$T_v = T(1.0 + 0.608 q_v), \tag{4.2}$$

where T is the temperature and q_r is the specific humidity, and the Exner function,

$$\pi = \left(\frac{p}{p_{00}} \right)^{R_d / C_p}, \tag{4.3}$$

where C_p is the specific heat at constant pressure, and p_{00} the reference pressure.

The following variables can be decomposed into mean state and perturbation quantities such that

$$u(x,y,z,t) = \bar{u}\,(z) + u'(x,y,z,t), \tag{4.4}$$

$$v(x,y,z,t) = \bar{v}\,(z) + v'(x,y,z,t), \tag{4.5}$$

$$w(x,y,z,t) = \bar{w}\,(z) + w'(x,y,z,t), \tag{4.6}$$

$$\pi(x,y,z,t) = \bar{\pi}(z) + \pi'(x,y,z,t), \tag{4.7}$$

$$\theta(x,y,z,t) = \bar{\theta}(z) + \theta'(x,y,z,t), \tag{4.8}$$

where u, v, w are the Cartesian velocity components, θ is the potential temperature, t is time, and x, y, z are the Cartesian dimensions. The overbarred variables represent the time-invariant mean state, and the primed variables are the deviations. The mean state is assumed to be in hydrostatic balance such that

$$\frac{\partial \bar{\pi}}{\partial z} = -\frac{g}{C_p \bar{\theta}_v}, \tag{4.9}$$

where g is the acceleration due to gravity and $\bar{\theta}_v$ is the virtual potential temperature of the mean state

$$\bar{\theta}_v = \bar{\theta}(1.0 + 0.608 \bar{q}_v), \tag{4.10}$$

where $\bar{q}_v$ is the mean state water vapor.

The transformation of the vertical coordinate following Gal-Chen and Somerville (1975) is applied to map the lowest coordinate surface to an irregular lower boundary

$$\sigma = z_{top}\left(\frac{z - z_{sfc}}{z_{top} - z_{sfc}}\right), \tag{4.11}$$

where z_{top} is the depth of the model domain and z_{sfc} is the height of the topography. The basic set of model equations for the dynamical prognostic equations can be written as

$$\frac{\partial u}{\partial t} + C_p\theta_v\left(\frac{\partial \pi'}{\partial x} + G_x\frac{\partial \pi'}{\partial \sigma}\right) - K_D\left(\frac{\partial D_3}{\partial x} + G_x\frac{\partial D_3}{\partial \sigma}\right) = -u\left(\frac{\partial u}{\partial x}\right)_\sigma - v\left(\frac{\partial u}{\partial y}\right)_\sigma - \dot{\sigma}\left(\frac{\partial u}{\partial \sigma}\right) + fv + D_u + K_H\nabla^4 u, \tag{4. 12}$$

$$\frac{\partial v}{\partial t} + C_p\theta_v\left(\frac{\partial \pi'}{\partial y} + G_y\frac{\partial \pi'}{\partial \sigma}\right) - K_D\left(\frac{\partial D_3}{\partial y} + G_y\frac{\partial D_3}{\partial \sigma}\right) = -u\left(\frac{\partial v}{\partial x}\right)_\sigma - v\left(\frac{\partial v}{\partial y}\right)_\sigma - \dot{\sigma}\left(\frac{\partial v}{\partial \sigma}\right) - fu + D_v + K_H\nabla^4 v, \tag{4.13}$$

$$\begin{aligned}\frac{\partial w}{\partial t} + C_p\theta_v G_z\frac{\partial \pi'}{\partial \sigma} - K_D G_z\frac{\partial D_3}{\partial \sigma} = g\left(\frac{\theta'}{\overline{\theta}} + 0.608q_v' - q_c - q_r - q_s - q_i - q_g\right) \\ -u\left(\frac{\partial w}{\partial x}\right)_\sigma - v\left(\frac{\partial w}{\partial y}\right)_\sigma - \dot{\sigma}\left(\frac{\partial w}{\partial \sigma}\right) + D_w + K_H\nabla^4 w,\end{aligned} \tag{4.14}$$

$$\frac{\partial \pi'}{\partial t} + \frac{\overline{c}^2}{C_p\overline{\rho}\overline{\theta}_v^2}(D_3) = -u\left(\frac{\partial \pi'}{\partial x}\right)_\sigma - v\left(\frac{\partial \pi'}{\partial y}\right)_\sigma - \dot{\sigma}\left(\frac{\partial \pi'}{\partial \sigma}\right) - \frac{R_d\overline{\pi}}{C_v}\nabla_3\cdot\mathbf{V} + \frac{\overline{c}^2}{C_p\theta_v^2}\frac{d\theta_v}{dt}, \tag{4.15}$$

$$\frac{\partial \theta}{\partial t} = -u\left(\frac{\partial \theta}{\partial x}\right)_\sigma - v\left(\frac{\partial \theta}{\partial y}\right)_\sigma - \dot{\sigma}\left(\frac{\partial \theta}{\partial \sigma}\right) + \frac{Q_\theta}{\overline{\rho}} + D_\theta + K_H\nabla^4(\theta - \overline{\theta}), \tag{4.16}$$

where C_v is the specific heat at constant volume, f is the Coriolis parameter, Q_θ is the source and sink of heat, and q_c, q_r, q_i, q_s, and q_g are the mixing ratios of cloud water, rain water, cloud ice, snow, and graupel, respectively. The density and potential temperature weighted three-dimensional divergence D_3 is defined as

$$D_3 = \left(\frac{\partial}{\partial x} + G_x\frac{\partial}{\partial \sigma}\right)(\overline{\rho}\overline{\theta}_v u) + \left(\frac{\partial}{\partial y} + G_y\frac{\partial}{\partial \sigma}\right)(\overline{\rho}\overline{\theta}_v v) + G_z\frac{\partial(\overline{\rho}\overline{\theta}_v w)}{\partial \sigma}. \tag{4.17}$$

The speed of sound for the mean state C can be expressed as

$$\overline{c} = \left(\frac{C_p R_d\overline{\pi}\overline{\theta}_v}{C_v}\right)^{1/2}. \tag{4.18}$$

The metrics for the coordinate transformation are defined as

$$G_x = \frac{\partial \sigma}{\partial x} = \left(\frac{\sigma - z_{top}}{z_{top} - z_{sfc}}\right)\left(\frac{\partial z_{sfc}}{\partial x}\right), \tag{4.19}$$

$$G_y = \frac{\partial \sigma}{\partial y} = \left(\frac{\sigma - z_{top}}{z_{top} - z_{sfc}}\right)\left(\frac{\partial z_{sfc}}{\partial y}\right), \tag{4.20}$$

$$G_z = \frac{\partial \sigma}{\partial z} = \frac{z_{top}}{z_{top} - z_{sfc}}, \tag{4.21}$$

$$\dot{\sigma} = G_x u + G_y v + G_z w. \tag{4.22}$$

The terms D_u, D_v, D_w, and D_θ represent the subgrid-scale mixing and are represented following Mellor and Yamada (1982) and Yamada (1982). The terms involving the diffusion coefficient K_D represent divergence damping that control the amplification of sound waves (Skamarock and Klemp 1992). The coefficient for fourth-order accurate diffusion K_H is used to control nonlinear instability. For more information about nonlinear instability see Haltiner and Williams (1980).

The equations are solved on a staggered, scheme C grid (Arakawa and Lamb 1977). The computational grid features: (1) the *u*-component are one-half grid intervals between the mass points in the x direction, (2) the v-component are one-half grid intervals between the mass points in the y direction, and (3) the *w*-component, are coincident with the mass variables. In general, all derivatives are computed to second-order accuracy. Exceptions are the horizontal diffusion and an option for fourth-order accurate horizontal advection. By using fourth-order accurate horizontal diffusion, the damping is much more specific to the removal of high-frequency modes.

Solving the fully compressible, nonhydrostatic equations explicitly is extremely computational expensive because of the presence of sound waves that severely limit the time step required to maintain computational stability. One approach for addressing this problem is to treat the sound wave modes separately on a small time step following Klemp and Wilhelmson (1978) and Skamarock and Klemp (1992). To apply this technique, the terms that govern the sound waves are isolated in the dynamical equations and collected on the left-hand side of the equations. For a small time step $\Delta\tau_a$ and a large time step Δt_a, Equations (4.12) through (4.15) are integrated as

$$u^{t-\Delta t_a+n\Delta\tau_a} = u^{t-\Delta t_a+(n-1)\Delta\tau_a} - \Delta\tau_a\left[C_p\overline{\theta}_v\left(\frac{\partial\pi'}{\partial x}\right)^{t-\Delta t_a+(n-1)\Delta\tau_a} + K_D\left(\frac{\partial D_3}{\partial x}\right)^{t-\Delta t_a+(n-1)\Delta\tau_a} + RHS_u^t\right], \tag{4.23}$$

$$v^{t-\Delta t_a+n\Delta\tau_a} = v^{t-\Delta t_a+(n-1)\Delta\tau_a} - \Delta\tau_a\left[C_p\overline{\theta}_v\left(\frac{\partial\pi'}{\partial y}\right)^{t+\Delta t_a-(n-1)\Delta\tau_a} + K_D\left(\frac{\partial D_3}{\partial y}\right)^{t-\Delta t_a+(n-1)\Delta\tau_a} + RHS_v^t\right], \tag{4.24}$$

$$w^{t-\Delta t_a+n\Delta\tau_a} = w^{t-\Delta t_a+(n-1)\Delta\tau_a} - \Delta\tau_a\left[C_p\overline{\theta}_v\left(\frac{\partial\pi'}{\partial z}\right)^{t-\Delta t_a+(n-1)\Delta\tau_a} + K_D\left(\frac{\partial D_3}{\partial z}\right)^{t-\Delta t_a+(n-1)\Delta\tau_a} + RHS_w^t\right], \tag{4.25}$$

$$(\pi')^{t-\Delta t_a+n\Delta\tau_a} = (\pi')^{t-\Delta t_a+(n-1)\Delta\tau_a} - \Delta\tau_a\left[\left(-\frac{R_d\overline{\pi}}{C_v}\nabla_3\cdot\mathbf{V}\right)^{t-\Delta t_a+n\Delta\tau_a} + RHS_{(\pi')}^t\right], \tag{4.26}$$

where RHS_u, RHS_v, RHS_w, and $RHS_{(\pi)}$ are the right-hand side terms in Equations (4.12), (4.13), (4.14), and (4.15), respectively. Equations (4.23) and (4.24) are stepped forward each small time step. Equations (4.25) and (4.26) are solved simultaneously by inverting a tri-diagonal matrix in the vertical to obtain *w*. For stability, $\Delta\tau_a$ must satisfy

$$\Delta\tau_a < \frac{1}{c}\left\{\frac{1}{x^2}+\frac{1}{y^2}\right\}^{-1/2} \tag{4.27}$$

(Skamarock and Klemp 1992). In the COAMPS model code, $\Delta\tau_a$ is represented by *mtaua*. More details regarding the stability of this method are found in Skamarock and Klemp (1992).

The equations are solved using the centered-in-time or leapfrog scheme (Haltiner and Williams 1980). Typically, the time splitting due to the leapfrog scheme is not problematic (e.g., Klemp and Wilhelmson 1978). However, a Robert (1966) time smoother is applied to assure that any tendencies that may tend to decouple the odd and even time steps are stable. For any variable ϕ, the Robert time filter is applied as

$$\left.\begin{aligned} \phi^{*t+\Delta t} &= \phi^{t-\Delta t} + 2\Delta t\phi^{*t} \\ \phi^{t} &= \phi^{*t} + \alpha\left(\phi^{*t+\Delta t} - 2\phi^{*t} + \phi^{t-\Delta t}\right) \end{aligned}\right\}. \tag{4.28}$$

The first equation corresponds to the leapfrog step for $\partial\phi/\partial t = F$, with the asterisk corresponding to provisional terms that have not yet been smoothed through application of the second step. The net effect is to produce strong damping of the computational mode, while the physical mode is generally not affected (Asselin 1972). For typical COAMPS applications, α is 0.2.

4.2 Moist Physics

The moist physics scheme in COAMPS currently consists of a single-moment bulk prediction of mixing ratio developed by Rutledge and Hobbs (1983), hereafter referred to as RH83. The RH83 scheme is a bulk cloud microphysical model based on the Lin et al. (1983) formulation, with single-moment prediction of mixing ratio for five microphysical variables (vapor, pristine ice, snow, rain, and cloud water). The primary assumptions used in this scheme include use of the Marshall and Palmer (1948) size distribution, the Kessler (1969) autoconversion, and the Fletcher (1962) formulation for the nucleation of pristine ice. Terminal velocities are computed for the rain and snow fields while the remaining fields are treated as scalar tracers.

The bulk scheme is invoked after the model dynamics have been calculated and the scalar prognostic variables have been updated for advection, diffusion, and mixing processes. The updates are performed in the main microphysics driver subroutines *aforqx.F*, *amixtq.F* (implicit mixing of theta and vapor), and *amixnf.F* (implicit mixing of scalar fields). The primary difference between *amixtq.F* and *amixnf.F* is that surface fluxes are used as a lower boundary condition in the implicit calculation for theta and vapor in *amixtq.F*. Updates to the microphysical scalars in *aforqx.F* differ slightly, depending on the value of the model grid spacing and whether the microphysical variable has a calculated fallspeed (as with the rain and snow fields). For rain and snow fields and grid spacings greater than *dxmeso* (a *coamnl* namelist variable defines the grid resolution in kilometers, where below it the cumulus parameterization is turned off and above it the advection of rain and snow is turned off); the only update to the mixing ratio tendency arises from the precipitation flux. This parameter is treated with a forward/upstream time differencing technique and time splitting for numerical stability. For all other conditions, the mixing ratio for a given species is updated for horizontal mixing, horizontal diffusion, advection, and the precipitation fluxes if applicable. The implicit vertical mixing is subsequently computed in *amixtq.F* and *amixnf.F* using the updated microphysical fields obtained in *aforqx.F*.

4.2.1 Single-moment bulk scheme

After the microphysical variables, theta, and model dynamics have been updated, the code proceeds to calculate the updates from the model physics. The updates in the case of the explicit moist physics occur within the routine *adjtq.F*. All of the constants are first computed in the routine *mic_cnst.F*. Next, the pressure, temperature, and saturation values required by the routine *adjtq.F* are computed over the entire grid. To avoid unnecessary microphysics calculations, a check is performed to verify if there is either active condensate at the grid point or the grid point is supersaturated. If either of these conditions is satisfied, the active points within a given *i-k* slab (an east-west plane) are then collected and passed into the routine *adjtq.F* along with the previously calculated values of pressure, temperature, and saturation, for further adjustment.

Within the microphysics scheme itself, updates are made to temperature, vapor, and condensate fields by using a newly formed bulk moist physics scheme based on the work of Rutledge and Hobbs (1983) (RH83); Rutledge and Hobbs (1984) (RH84); Khairoutdinov and Kogan (2000); Meyers et al. (1992) and (1997); Cotton and Anthes (1989); Reisner et al. (1998); Soong and Ogura (1973); and Hallet and Mossop (1974). In the original bulk (RH83) scheme used in COAMPS, the updates were performed on a process-by-process basis, with sequential updates made to the vapor, condensate, or temperature fields. This adjustment scheme was found to produce cycling between subsaturated and super-saturated conditions during the execution of *adjtq.F* as each process was computed. The RH83 scheme was also originally designed for explicit simulations of the "seeder-feeder" mechanism in a highly idealized environmental setting. As such, the scheme was designed to account for the growth of a specified pristine ice field into snow, and through settling, for the development of rain and cloud water below the melting level. Other than riming of snow by the cloud liquid water, no other interactions between the cloud/rain water and ice species were allowed to occur in the original scheme. The desire to model other cloud types where conversions between the rain and cloud water with the ice phase, outlined below, led to consideration of several upgrades and modifications to the original RH83 scheme.

For the most part, the upgrades mentioned here reflect changes in cloud microphysics schemes that have appeared in the literature since the RH83 scheme was originally published. In addition to the five scalar fields of the original scheme, predictive equations for graupel (heavily rimed ice crystal or frozen drop) (Rutledge and Hobbs (1984), hereafter referred to as RH84) and drizzle (Khairoutdinov and Kogan (2000), hereafter referred to as the KK2000 scheme) have been added. These schemes have been included to improve the prediction of coastal stratus, orographic clouds, and convective systems.

The RH84 graupel scheme improves the liquid/ice conversions that were lacking in the original RH83 scheme. In addition, other aspects of the ice prediction have been modified to include the Meyers et al. (1992) ice nucleation, the nucleation formula discussed by Cooper and Haines (1986), homogeneous freezing, ice multiplication processes (Hallet and Mossip 1974), and by incorporating a non zero fall speed for the pristine ice field. These adjustments made to the ice nucleation are found in the routine *conice.F*. The development of a full two-moment ice microphysics scheme is in progress and will be based on papers by Meyers et al. (1997) and Reisner et al. (1998).

4.2.2 Adjustment to saturation scheme

The sequential adjustment to saturation scheme originally used in RH83 was found to have a strong influence on the modeled cloud fields and has subsequently been replaced by an alternate approach. It was found that updating the temperature and vapor after a given process (the initiation of ice, for example) led to a cycling between supersaturated and subsaturated conditions at a given grid point during a single timestep.

The undesired cycling was removed. The sequential updating has been retained to some extent. Microphysical processes have been grouped into three primary categories involving either pure temperature changes (treated first in routine *adjtq.F*), vapor and temperature changes (treated second), and collection and autoconversion (treated last). The terms involving only temperature changes are calculated in the routines *frz.F* (homogeneous freezing of cloud and rain water at –40 C); *eqa25a.F* (melting of snow); *eqa28.F* (melting of pristine ice); *eqa11g.F* (collection of cloud water by graupel); *eqa27r.F* (collection of cloud water by pristine ice); *eqa7g.F* (pristine ice collecting rain); *eqa8g.F* (snow collecting ice); *eqa9g.F* (rain collecting snow); *eqa13g.F* (rain collecting graupel); *eqa18g.F* (melting graupel); *eqa21g.F* (graupel accreting cloud water); and *eqa22g.F* (enhanced graupel melting due to accreting cloud water). The routine names follow the previous naming convention in which the routine is named after the equation appearing in either RH83, RH84 (includes the *eqaxxg.F* syntax), or Reisner (*eqaxxr.F*). Each tendency computed is limited by the total value of the field being adjusted in each of these routines.

After the temperature terms, the slope parameters (*slope.F*), and the graupel, rain, snow, and pristine ice fall speeds (routines *tgqg.F*, *tgqr.F*, *tgqs.F*, *tgqi.F*, respectively) are computed, the temperature is updated (routine

adjmlt.F). In this routine, each sink term is normalized, as appropriate, to prevent any negative mixing ratio values from developing in the model. The normalized rates are then used to update the microphysical variables and temperature. The saturation values over liquid and ice are then recomputed (routine *qsatvi.F*) as well as the slope parameters. This accounts for any changes in the temperature or mixing ratio terms arising in *adjmlt.F*.

The second stage in the adjustment process is to compute the tendencies affecting both vapor and temperature. The current adjustment follows one of two separate procedures based upon the work of Asai (1964) and Soong and Ogura (1973). Both approaches represent implicit techniques to obtain the maximum vapor and temperature changes that are allowed to occur during the time step for liquid clouds only.

Current work by Tao (1989) uses an ice/liquid mass weighted saturation value such that, in the absence of liquid, the scheme adjusts toward the ice saturation value at temperatures below 273 K. The adjustment is performed twice in COAMPS — once toward the saturation value with respect to liquid at all levels, and then again toward the ice saturation value. The adjustment for the liquid phase follows from recognizing that the Clapyreon equation and thermodynamic equation may be written to first order as

$$q_{vs}^{\tau+1} - q_{vs}^{*} = \frac{L_v q_{vs}^{*}}{R_v T^2}(T^{\tau+1} - T^{*}), \tag{4.29}$$

$$T^{\tau+1} - T^{*} = \frac{L_v}{c_p}(q_v^{*} - q_{vs}^{\tau+1}). \tag{4.30}$$

In these equations, the starred quantities represent the values of the vapor and temperature fields after the model dynamics and mixing have been computed. Substituting from Equation (4.30) into (4.29) and solving for the new saturation value gives

$$q_{vs}^{\tau+1} = \frac{q_{vs}^{*}\left(1 + \dfrac{L_v^2 q_v^{*}}{c_p R_v T^2}\right)}{\left(1 + \dfrac{L_v^2 q_{vs}^{*}}{c_p R_v T^2}\right)}. \tag{4.31}$$

By using this expression, Equation (4.30) gives

$$T^{\tau+1} - T^{*} = \frac{L_v(q_v^{*} - q_{vs}^{*})}{c_p\left(1 + \dfrac{L_v^2 q_{vs}}{c_p R_v T^2}\right)}. \tag{4.32}$$

Similar expressions are written for the ice phase adjustment after substituting for the appropriate latent heat and ice saturation values.

Updates to vapor and temperature for the liquid phase are computed first in the routine *nrmtqw.F*. The update computations require the terms: *eqa6.F* (condensation/evaporation of cloud water); *eqa12.F* (condensation/evaporation of rain; *eqa27.F* (evaporation of melting snow); and *eqa19g* (evaporation of melting graupel). The implicit adjustment scheme (routine *qtadj.F*) calculates the maximum allowable vapor change over a time step at which the model would be exactly at a state of saturation for a given grid point after all the processes have been taken into account.

The key variable computed in *qtadj.F* is the array *qadj* (i.e., $q_{vs}^{\tau+1}$ given above). This array holds the value of the saturation vapor mixing ratio derived from the implicit scheme. As such, it represents the final value of the vapor

mixing ratio toward which the vapor tendency is driven through the adjustment procedure. This value differs, depending on whether the desired saturation value is for ice or liquid.

The available vapor for the timestep is then *qvtemp(i,k)-qadj(i,k)* (*qvtemp(i,k)-qsatv(i,k)* in the old scheme). The vapor difference is used to calculate all of the rate equations. If the air is subsaturated, a check is performed to test if the available condensate will provide a sufficient source of vapor such that the saturation value can be obtained. If the condensate is sufficient, a limit is placed on the amount of evaporation/sublimation that is allowed to occur during the time step. All condensate values that are less than the limit are allowed to evaporate/sublimate completely during the time step.

In the routine *nrmtqw.F*, each microphysical species is given equal access to the available vapor or condensate excess at the beginning of the time step. All rates within each group are then normalized to prevent negative values of any of the positive definite fields. Subjective evaluation of model output indicates that these steps have led to significant improvement in the model cloud predictions, particularly in the pristine ice phase that suffered as a result of the cycling in the saturation field. The routine *nrmtqw.F* also updates the cloud and rain concentration for the KK2000 drizzle parameterization based on the normalized rates computed in the routine. Finally, all condensate values are checked to ensure positive definiteness, and any values less than the threshold (*pcut*) are set to zero.

After the call to *nrmtqw.F*, the saturation values and slope factors are recomputed and the routine *qtadj.F* is called again to compute *qadj* for the subsequent ice phase calculations. The terms involving vapor and temperature changes in the ice phase are updated in the routine *nrmtqi.F* and include the following source terms: *eqa15.F* (initiation of pristine ice); *eqa18.F* (depositional growth of pristine ice); *eqa26.F* (depostional growth of snow); and *eqa17g.F* (depositional growth of graupel). As in *nrmtqw.F*, the microphysical rates are normalized, with each process having full access to the available vapor at this stage in the routine. Note that the model is being driven toward ice saturation at temperatures less than 0° C.

After the calls to *adjmlt.F*, *nrmtqw.F*, and *nrmtqi.F*, the temperature and vapor updates are complete. Before the final update to these and other microphysical variables is performed, adjustments are made to the condensate to fields account for autoconversion and various collection terms. The updates occur in the routine *nrmcol.F* and include the following processes: *eqa7.F* (autoconversion of cloud water to rain water); *eqa9.F* (collection of cloud water by rain water); *eqa19.F* (conversion of cloud ice to snow); *eqa10g.F* (loss of snow due to collisions with cloud water); *eqa21.F* (collection of cloud ice by snow); *eqa5g.F* (collection of cloud ice by rain); *eqa12g.F* (collection of cloud ice by graupel); *eqa14g.F* (collection of snow by graupel); and *eqa20g.F* (shedding of accreted water). As in the other routines, the rates have been normalized to prevent negative tendencies in any of the positive definite microphysical variables. After *nrmcol.F*, the final step is updating the vapor temperature and condensate fields by scattering the compressed fields back to the full three-dimensional model grids.

4.2.3 Drizzle parameterization

One weakness in the RH83 scheme is the lack of cloud condensation nuclei and the relatively arbitrary choice for the Kessler autoconversion threshold. As discussed by Cotton and Anthes (1989), it is highly desirable to have a scheme that naturally accounts for differences in oceanic and continental aerosol sources when attempting to model regional cloud differences. The KK2000 drizzle parameterization has recently been implemented in COAMPS (the initial implementation of this scheme into COAMPS was performed by Dr. David Mechem, University of Oklahoma) to address this issue. This scheme uses a large eddy simulation (LES)-based approach to determine the autoconversion rates and allows for a more sophisticated coupling between the modeled aerosol and cloud fields than previously available in COAMPS. The scheme requires additional prognostic equations for cloud and rain water number concentration as well as ambient aerosol number concentration. As such, it will likely serve as the basis of the liquid phase of the full two-moment bulk microphysical scheme in COAMPS.

The aerosol concentration in the original KK2000 scheme was set as a user-specified constant that applied over the entire domain. This constraint has been relaxed somewhat in the current implementation by allowing the user to specify both horizontally and vertically varying aerosol concentrations. A different vertical profile, for example, may be specified for the land and ocean grid points. This change will allow simplified tests to be conducted on the impact of these separate aerosol distributions on the evolving cloud field in COAMPS within the coastal zone. Once specified, the aerosols are subject to advection, mixing, and precipitation fallout. Currently, additional source terms for the aerosol are not specified and thus, as pointed out by KK2000, the overall concentration may fall throughout the simulation in the presence of drizzle (one aerosol per droplet). Fully coupling the drizzle scheme to an analyzed aerosol field is under consideration but faces significant challenges. For now, the work will focus on improving the interaction between the drizzle and aerosol by including additional source/sink terms for the aerosol that might arise from surface fluxes or more advanced cloud scavenging processes. Coupling this scheme to a full two-moment mixed-phase microphysics scheme is also in progress.

4.2.4 Equations

Equations for the microphysics module of COAMPS are provided in the following sections. Table 4.1 summarizes the equation symbols.

4.2.4.1 Size distributions

$$N_{DR} = N_{0R}\exp(-\lambda_R D_R)dD_R \qquad \textit{Rain} \qquad (4.33)$$

$$N_{DS} = N_{0S}\exp(-\lambda_S D_S)dD_S \qquad \textit{Snow} \qquad (4.34)$$

$$N_{DG} = N_{0G}\exp(-\lambda_G D_G)dD_G \qquad \textit{Graupel} \qquad (4.35)$$

$$M_0 = 10^{-12}Kg \quad (D_0 = 12.9\mu\text{m}) \qquad \textit{Ice} \qquad (4.36)$$

$$N_c = \text{const}; \quad r_c = \left(\frac{3q_c\rho_0}{4\pi\rho_L N_c}\right)^{1/3} \qquad \textit{Cloud Water} \qquad (4.37)$$

4.2.4.2 Slope factors

$$\lambda_R = \left(\frac{\pi\rho_L N_{0R}}{\rho q_r}\right)^{0.25} \qquad \textit{Rain} \qquad (4.38)$$

$$\lambda_S = \left(\frac{\pi\rho_S N_{0S}}{\rho q_s}\right)^{0.25} \qquad \textit{Snow} \qquad (4.39)$$

$$\lambda_G = \left(\frac{\pi\rho_G N_{0G}}{\rho q_g}\right)^{0.25} \qquad \textit{Graupel} \qquad (4.40)$$

4.2.4.3 Mass/diameter relationships

$$M(D_R) = \frac{\pi\rho_L D_R^3}{6} \qquad \textit{Rain} \qquad (4.41)$$

$$M(D_I) = 0.06135 D_I^2 \qquad \textit{Pristine Ice} \tag{4.42}$$

$$M(D_S) = \frac{\pi \rho_S D_S^3}{6} \qquad \textit{Snow} \tag{4.43}$$

$$M(D_G) = \frac{\pi \rho_G D_G^3}{6} \qquad \textit{Graupel} \tag{4.44}$$

4.2.4.4 Fall speeds

Rain: RH83 (A1,A2,A3): Subroutine tvqr.F

NOTE: slope factor should be in cm for this calculation

$$\bar{V}_R = \frac{\int_0^\infty N_{DR}(D_R) M(D_R) V_R(D_R) dD_R}{\int_0^\infty N_{DR}(D_R) M(D_R) dD_R} \tag{4.45}$$

$$V_R(D_R) = -0.267 + 51.5 D_R - 102.25 D_R^2 + 75.5 D_R^3 \tag{4.46}$$

$$\bar{V}_R = (-0.267 + 206 \lambda_R^{-1} - 2.045 \times 10^3 \lambda_R^{-2} + 9.06 \times 10^3 \lambda_R^{-3}) \left(\frac{p_0}{p} \right)^{0.4} \tag{4.47}$$

Snow: RH83 (A1,A4,A5): Subroutine tvqs.F

$$\bar{V}_S = \frac{\int_0^\infty N_{DS}(D_S) M(D_S) V_S(D_S) dD_S}{\int_0^\infty N_{DS}(D_S) M(D_S) dD_S} \tag{4.48}$$

$$V_S(D_S) = a'' D_S^b \left(\frac{p_0}{p} \right)^{0.4} \tag{4.49}$$

$$\bar{V}_S = a'' \frac{\Gamma(4+b)}{6} \lambda_S^{-b} \left(\frac{p_0}{p} \right)^{0.4} \tag{4.50}$$

Graupel: RH84 (A1, A2, A3): Subroutine tvqg.F

$$\bar{V}_G = \frac{\int_0^\infty N_{DG}(D_G) M(D_G) V_G(D_G) dD_G}{\int_0^\infty N_{DG}(D_G) M(D_G) dD_G} \tag{4.51}$$

$$V_G(D_G) = \bar{a} D_G^{\bar{b}} \left(\frac{p_0}{p} \right)^{0.4} \tag{4.52}$$

$$\bar{V}_G = \bar{a} \frac{\Gamma(4+\bar{b})}{6} \lambda_G^{-\bar{b}} \left(\frac{p_0}{p} \right)^{0.4} \tag{4.53}$$

Pristine Ice: CA89 (4.62): Subroutine tvqi.F

$$\overline{V}_I = 304 D_I \left(\frac{p_0}{p} \right)^{0.5} \tag{4.54}$$

4.2.4.5 Pristine ice nucleation

Primary Nucleation (Fletcher 1962) RH83 (A13): Subroutine conice.F (icon=1)

$$n_c = n_0 \exp[\beta(T_0 - T)] \tag{4.55}$$

Primary and Secondary Nucleation (WCMH95, CTRM85, and MDC92): Subroutine conice.F (icon=2)

Prognostic Concentration: WCMH95 (65)

$$\Delta N_t = \left(N_t\right)_d + \left(N_t\right)_c + \left(N_t\right)_s + \left[\left(\frac{dN_t}{dt} \right)_v + \left(\frac{dN_t}{dt} \right)_t + \left(\frac{dN_t}{dt} \right)_b \right] \Delta t \tag{4.56}$$

Depostion/freezing: MDC92 (2.4)

$$\left(N_t\right)_d = \exp\left\{ a + b\left[100\left(S_i - 1\right) \right] \right\} \tag{4.57}$$

Contact Freezing: MDC92 (2.6)

$$\left(N_t\right)_c = \exp\left\{ a + b\left(T_0 - T_c\right) \right\} \tag{4.58}$$

Secondary Ice production: CTRM86 (82)

$$\left(N_t\right)_s = \frac{1}{\rho_0} \left[\beta_5 \rho_0 + 4 \times 10^{-3} \frac{N_c}{\bar{r}_c} f_3\left(m_c\right) \right] * \left[f_1\left(T_i\right) * piacw + f_1\left(T_g\right) * pgacw \right] \tag{4.59}$$

where: CTRM86 (81,77,72,79)

$$f_3(m_c) = \begin{Bmatrix} 0, & m_c < 1.26 \times 10^{-6} g \\ 2.27 \ln m_c + 13.39, & 1.26 \times 10^{-6} g \le m_c \le 3.55 \times 10^{-6} g \\ 1, & m_c > 3.55 \times 10^{-6} g \end{Bmatrix} \tag{4.60}$$

$$m_c = \frac{q_c \rho_0}{N_c} \tag{4.61}$$

$$f_1(T_p) = \begin{Bmatrix} 0, & T_p > 270.16 \\ \left[\frac{T_p - 268.16}{2} \right], & 270.16 \ge T_p \ge 268.16 \\ \left[\frac{T_p - 265.16}{3} \right], & 268.16 \ge T_p \ge 265.16 \\ 0, & 265.16 \ge T_p \end{Bmatrix} \tag{4.62}$$

$$N_{12} = N_c f_3(m_3) \tag{4.63}$$

Diffusiophoresis: WCMH (52)

$$\left(\frac{dN_t}{dt}\right)_v = F_1 F_2 \frac{R_v T_a}{L_{lv}\rho_a} \tag{4.64}$$

Thermophoresis: WCMH (53)

$$\left(\frac{dN_t}{dt}\right)_t = F_1 F_2 \frac{f_t}{\rho_a} \tag{4.65}$$

Brownian Motion: WCMH (54,55,56)

$$\left(\frac{dN_t}{dt}\right)_b = F_1 \varphi_a \tag{4.66}$$

$$F_1 = 2\pi D_c N_{tc} N_a \tag{4.67}$$

$$F_2 = \frac{\kappa}{p}(T_{ac} - T_{cc}) \tag{4.68}$$

where: WCMH (57,58,59)

$$f_t = \frac{0.4\left[1+1.45K_n + 0.4K_n \exp\left(\frac{-1}{K_n}\right)\right](\kappa + 2.5K_n\kappa_a)}{(1+3K_n)(2\kappa + 5\kappa_a K_n + \kappa_a)} \tag{4.69}$$

$$K_n = \frac{\lambda_{a0} T p_0}{T_0 p R_a} \tag{4.70}$$

$$\psi_a = \frac{kT_{cc}}{6\pi R_a \mu}(1+K_n) \tag{4.71}$$

Diagnostic Crystal Concentration: HR85 (3) (icon=3)

$$(N_t) = \left(\frac{D_T}{18.5}\right)^{8.4} \tag{4.72}$$

Diagnostic Crystal Concentration (Cooper and Haines, 1986): (icon=4)

$$N_t = 0.005 * l2m3 * \exp(0.304 * (T - T_{\min})) \tag{4.73}$$

4.2.4.6 Saturation adjustment: SO73/ARPS: Subroutine sat_adj.F

Teten's Formula: ARPS (6.3.19,6.3.20)

$$q_{vs} = \frac{380}{p}\exp\left(a_w \frac{T - 273.16}{T - b_w}\right) \tag{4.74}$$

where

$$a_w = \begin{Bmatrix} 17.27 & T \geq 273.16 \\ 21.875 & T < 273.16 \end{Bmatrix} \tag{4.75}$$

$$b_w = \begin{Bmatrix} 35.5 & T \geq 273.16 \\ 7.5 & T < 273.16 \end{Bmatrix} \tag{4.76}$$

First , adjust cloud and rain water for all T, then update T.

Vapor adjustment: ARPS(6.6.6): Subroutine sat_adj.F

$$\delta q_{vs} = \frac{-\left[q_v^* - q_{vs}^*\right]}{1 + \dfrac{a_w (273.15 - b_w) q_{vs}^* \dfrac{L_v}{c_p}}{\left[T^* - b_w\right]^2}} \tag{4.77}$$

Recompute for ice calculations for T<273.16.

$$\delta q_{vs} = \frac{-\left[q_v^* - q_{vsi}^*\right]}{1 + \dfrac{a_w (273.15 - b_w) q_{vsi}^* \dfrac{L_{vi}}{c_p}}{\left[T^* - b_w\right]^2}} \tag{4.78}$$

Temperature adjustment:

$$\delta T = \begin{Bmatrix} -\dfrac{L_v}{c_p}\delta q_{vs} & (T \geq 273.16) \\ -\dfrac{L_s}{c_p}\delta q_{vs} & (T < 273.16) \end{Bmatrix} \tag{4.79}$$

4.2.4.7 Conservation equations

Water Vapor (q_v): RH84 (A23)

$$S_v = -\left[\begin{matrix} PCOND + PREVP + PSDEP + PMLTEV(T \geq 273.16) + PINT + PDEPI \\ + PGDEP + PMLTGE(T \geq 273.16) \end{matrix}\right] \tag{4.80}$$

Cloud Water (q_c): RH84 (A24)

$$S_c = PCOND + PSMLTI(T \geq 273.16) - PRAUT - PRACW - PSACW - PGACW - PSFW \tag{4.81}$$

Rain (q_r) where $(T \geq 273.16)$ *: RH84 (A26)*

$$\begin{aligned} S_r = {} & PRAUT + PRACW + PREVP - PGMLT - PSMLT - PGACRM - PGACWM + PSACW \\ & PRACS + PGSHR - PGACR \end{aligned} \tag{4.82}$$

Rain (q_r) where ($T \le 273.16$) *: RH84 (A27)*

$$S_r = PRAUT + PRACW + PREVP - PGACR - PSACR - PIACR \tag{4.83}$$

Snow (q_s) where ($T \ge 273.16$) *: RH84 (A28)*

$$S_s = PSMLT - PRACS - PGACS + PMLTEV \tag{4.84}$$

Snow (q_s) where ($T \le 273.16$) *: RH84 (A29)*

$$\begin{aligned} S_s = {} & PCONV + PSACI + PSDEP - PGACS + PSFW + PSFI + PRACI(\delta_1) + PIACR(\delta_1) \\ & PSACR(\delta_2) - PRACS(1-\delta_2) + PSACW(\delta_3) - PWACS(1-\delta_3) \end{aligned} \tag{4.85}$$

where: RH84 (A30, A31, A32)

$$\delta_1 = \begin{cases} 0 & if\ q_r > 0.1 g\,kg^{-1} \\ 1 & otherwise \end{cases} \tag{4.86}$$

$$\delta_2 = \begin{cases} 0 & if\ q_r\ and\ q_s > 0.1 g\,kg^{-1} \\ 1 & otherwise \end{cases} \tag{4.87}$$

$$\delta_3 = \begin{cases} 0 & if\ q_s > 0.1 g\,kg^{-1}\ and\ q_c > 0.5 g\,kg^{-1} \\ 1 & otherwise \end{cases} \tag{4.88}$$

Graupel (q_g) where ($T \ge 273.16$) *: RH84 (A33)*

$$S_g = PGMLT + PGACRM + PGACWM + PGACS + PMLTGE \tag{4.89}$$

Graupel (q_g) where ($T \le 273.16$) *: RH84 (A34)*

$$\begin{aligned} S_g = {} & PGACW + PGACR + PGACI + PGACS + PGDEP + PRACI(1-\delta_1) \\ & PIACR(1-\delta_1) + PSACR(1-\delta_2) + PRACS(1-\delta_2) + PSACW(1-\delta_3) + PWACS(1-\delta_3) \end{aligned} \tag{4.90}$$

Thermodynamics where ($T \ge 273.16$) *: RH84 (A35)*

$$\begin{aligned} S_h = {} & \frac{L_f}{c_p}\left[PGMLT + PSMLT + PGACWM + PGACRM - PSMLTI - PRACS\right] \\ & + \frac{L_v}{c_p}\left[PCOND + PREVP + PMLTEV + PMLTGE\right] \end{aligned} \tag{4.91}$$

Thermodynamics where ($T \le 273.16$) *: RH84 (A36)*

$$\begin{aligned} S_h = {} & \frac{L_f}{c_p}\left[PSACW + PIACR + PSACR + PGACR + PGACW\right] \\ & + \frac{L_f}{c_p}\left[PCOND + PREVP\right] \\ & + \frac{L_s}{c_p}\left[PGDEP + PSDEP + PDEPI + PINT\right] \end{aligned} \tag{4.92}$$

4.2.4.8 Source terms (S_0)

Condensation of cloud water: RH83 (A6) Subroutine eqa6.F

$$PCOND = \rho\left(q_v - q_{sw}\right)\left[\Delta t\left(1+\frac{L_v^2 q_{sw}}{c_p R_w T^2}\right)\right]^{-1} \tag{4.93}$$

Autoconversion of cloud water: RH83 (A7) Subroutine eqa7.F

$$PRAUT = \alpha\rho(q_c - q_0) \tag{4.94}$$

Accretion of cloud water by rain: RH83 (A9) Subroutine eqa9.F

$$PRACW = \frac{\pi}{4}\rho q_c E_{RC} N_{0R}\left(\frac{p_0}{p}\right)^{0.4} * \left[\frac{a_0\Gamma(3)}{\lambda_R^3}+\frac{a_1\Gamma(4)}{\lambda_R^4}+\frac{a_2\Gamma(5)}{\lambda_R^5}+\frac{a_3\Gamma(6)}{\lambda_R^6}\right] \tag{4.95}$$

Evaporation/Condensation of Rain: RH83 (A12) Subroutine eqa12.F

$$PREVP = \frac{2\pi N_{0R}(S-1)}{A'+B'} * \left[\frac{0.78}{\lambda_R^2}+0.31\frac{\left(a'\frac{\rho}{\mu}\right)^{0.5}}{\lambda_R^3}\Gamma(3)\left(\frac{p_0}{p}\right)^{0.2}\right] \tag{4.96}$$

where

$$A' = \frac{L_v}{K_a T}\left(\frac{L_v M_w}{R^* T}-1\right) \quad \text{and} \quad B' = \frac{R^* T}{\chi M_w e_s} \tag{4.97}$$

Initiation of cloud ice: RH83 (A14) Subroutine eqa14.F

$$PINT = \frac{M_0 n_c}{\Delta t} \tag{4.98}$$

Depositional growth of cloud ice: RH83 (A18) Subroutine eqa18.F

$$PDEPI = \frac{4\overline{D}_I(S_i-1)n_c}{A''+B''} \tag{4.99}$$

where

$$A'' = \frac{L_s}{K_a T}\left(\frac{L_s M_w}{R^* T}-1\right) \tag{4.100}$$

$$B'' = \frac{R^* T}{\chi M_w e_{si}} \tag{4.101}$$

Conversion of cloud ice to snow: RH83 (A19) Subroutine eqa19.F

$$PCONV = \rho\left(\frac{q_i - q_{i\max}}{\Delta t}\right) \tag{4.102}$$

Collection of cloud ice by snow: RH83 (A20) Subroutine eqa21.F

$$PSACI = \frac{\rho\pi a'' q_i E_{SI} N_{0S}}{4}\left(\frac{p_0}{p}\right)^{0.4}\frac{\Gamma(b+3)}{\lambda_s^{b+3}} \tag{4.103}$$

Collection of cloud water by snow: RH83 (A22) Subroutine eqa22.F

$$PSACW = \frac{\rho\pi a'' q_c E_{SC} N_{0S}}{4}\left(\frac{p_0}{p}\right)^{0.4}\frac{\Gamma(b+3)}{\lambda_s^{b+3}} \tag{4.104}$$

Melting of Snow: RH83 (A25) Subroutine eqa25.F

$$PSMLT = \frac{-2\pi N_{0S}}{L_f}K_a(T-T_0)*\left[\frac{0.65}{\lambda_S^2}+0.44\left(\frac{a''\rho}{\mu}\right)^{0.5}\left(\frac{p_0}{p}\right)^{0.2}\frac{\Gamma\left(\frac{b}{2}+\frac{5}{2}\right)}{\lambda_s^{\frac{b}{2}+\frac{5}{2}}}\right] \tag{4.105}$$

Depositional growth of snow: RH83 (A26) Subroutine eqa26.F

$$PSDEP = \frac{4(S_i-1)N_{0S}}{A''+B''}*\left[\frac{0.65}{\lambda_S^2}+0.44\left(\frac{a''\rho}{\mu}\right)^{0.5}\left(\frac{p_0}{p}\right)^{0.2}\frac{\Gamma\left(\frac{b}{2}+\frac{5}{2}\right)}{\lambda_s^{\frac{b}{2}+\frac{5}{2}}}\right] \tag{4.106}$$

Evaporation of melting snow: RH83 (A27) Subroutine eqa27.F

$$PMLTEV = \frac{4(S-1)N_{0S}}{A'+B'}*\left[\frac{0.65}{\lambda_S^2}+0.44\left(\frac{a''\rho}{\mu}\right)^{0.5}\left(\frac{p_0}{p}\right)^{0.2}\frac{\Gamma\left(\frac{b}{2}+\frac{5}{2}\right)}{\lambda_s^{\frac{b}{2}+\frac{5}{2}}}\right] \tag{4.107}$$

Melting of cloud ice: RH83 (A28) Subroutine eqa28.F

$$PSMLTI = \frac{\rho q_i}{\Delta t} \tag{4.108}$$

GRAUPEL SCHEME:

Graupel initiation by rain collecting ice: RH84 (A5) Subroutine eqa5g.F

$$PRACI = \frac{\pi}{4}\rho q_i E_{RI} N_{0R}\left(\frac{p_0}{p}\right)^{0.4}*\left[\frac{a_0\Gamma(3)}{\lambda_R^3}+\frac{a_1\Gamma(4)}{\lambda_R^4}+\frac{a_2\Gamma(5)}{\lambda_R^5}+\frac{a_3\Gamma(6)}{\lambda_R^6}+\right] \tag{4.109}$$

Graupel initiation by ice collecting rain: RH84 (A7) Subroutine eqa7g.F

$$PIACR = \frac{\pi^2}{24}\rho_L n_{ci} E_{RI} N_{0R}\left(\frac{p_0}{p}\right)^{0.4}*\left[\frac{a_0\Gamma(6)}{\lambda_R^6}+\frac{a_1\Gamma(7)}{\lambda_R^7}+\frac{a_2\Gamma(8)}{\lambda_R^8}+\frac{a_3\Gamma(9)}{\lambda_R^9}+\right] \tag{4.110}$$

Graupel initiation by snow collecting rain: RH84 (A8) Subroutine eqa8g.F

$$PSACR = E_{SR}\pi^2\rho_L\left|\overline{V}_R - \overline{V}_S\right|N_{0R}N_{0S}\left(\frac{p_0}{p}\right)^{0.4} * \left[\frac{5}{\lambda_R^6\lambda_S} + \frac{2}{\lambda_R^5\lambda_S^2} + \frac{0.5}{\lambda_R^4\lambda_S^3}\right] \tag{4.111}$$

Graupel initiation by rain collecting snow: RH84 (A9) Subroutine eqa9g.F

$$PRACS = E_{SR}\pi^2\rho_L\left|\overline{V}_R - \overline{V}_S\right|N_{0R}N_{0S}\left(\frac{p_0}{p}\right)^{0.4} * \left[\frac{5}{\lambda_S^6\lambda_R} + \frac{2}{\lambda_S^5\lambda_R^2} + \frac{0.5}{\lambda_S^4\lambda_R^3}\right] \tag{4.112}$$

Graupel initiation: loss of snow due to collisions with cloud water: RH84 (A10) Subroutine eqa10g.F

$$PWACS = \frac{\overline{n}_c\rho_S\pi^2 a'' E_{SC}N_{0S}}{24}\left(\frac{p_0}{p}\right)^{0.4}\frac{\Gamma(b+6)}{\lambda_s^{b+6}} \tag{4.113}$$

Collection of cloud water by graupel: RH84 (A11) Subroutine eqa11g.F

$$PGACW = \frac{q_c\rho\ \pi\ \overline{a}E_{GC}N_{0G}}{4}\left(\frac{p_0}{p}\right)^{0.4}\frac{\Gamma(\overline{b}+3)}{\lambda_G^{\overline{b}+3}} \tag{4.114}$$

Collection of cloud ice by graupel: RH84 (A12) Subroutine eqa12g.F

$$PGACI = \frac{q_i\rho\ \pi\ \overline{a}E_{GI}N_{0G}}{4}\left(\frac{p_0}{p}\right)^{0.4}\frac{\Gamma(\overline{b}+3)}{\lambda_G^{\overline{b}+3}} \tag{4.115}$$

Collection of rain by graupel: RH84 (A13) Subroutine eqa13g.F

$$PGACR = E_{GR}\pi^2\rho_L\left|\overline{V}_G - \overline{V}_R\right|N_{0R}N_{0G}\left(\frac{p_0}{p}\right)^{0.4} * \left[\frac{5}{\lambda_R^6\lambda_G} + \frac{2}{\lambda_R^5\lambda_G^2} + \frac{0.5}{\lambda_R^4\lambda_G^3}\right] \tag{4.116}$$

Collection of snow by graupel: RH84 (A14) Subroutine eqa14g.F

$$PGACS = E_{GS}\pi^2\rho_S\left|\overline{V}_G - \overline{V}_S\right|N_{0G}N_{0S}\left(\frac{p_0}{p}\right)^{0.4} * \left[\frac{5}{\lambda_S^6\lambda_G} + \frac{2}{\lambda_S^5\lambda_G^2} + \frac{0.5}{\lambda_S^4\lambda_G^3}\right] \tag{4.117}$$

Depositional growth of graupel: RH84 (A17) Subroutine eqa17g.F

$$PGDEP = \frac{2\pi(\overline{S}_i - 1)N_{0G}}{A'' + B''} * \left[\frac{0.78}{\lambda_G^2} + 0.31\left(\frac{\overline{a}\rho}{\mu}\right)^{0.5}\left(\frac{p_0}{p}\right)^{0.2}\frac{\Gamma\left(\frac{\overline{b}}{2}+\frac{5}{2}\right)}{\lambda_G^{\frac{\overline{b}}{2}+\frac{5}{2}}}\right] \tag{4.118}$$

Melting of graupel: RH84 (A18) Subroutine eqa18g.F

$$PGMLT = \frac{-2\pi N_{0G}}{L_f} K_a (T - T_0) * \left[\frac{0.78}{\lambda_G^2} + 0.31 \left(\frac{\bar{a}\rho}{\mu} \right)^{0.5} \left(\frac{p_0}{p} \right)^{0.2} \frac{\Gamma\left(\frac{\bar{b}}{2} + \frac{5}{2} \right)}{\lambda_G^{\frac{\bar{b}}{2}+\frac{5}{2}}} \right] \tag{4.119}$$

Evaporation of melting graupel: RH84 (A19) Subroutine eqa19g.F

$$PMLTGE = \frac{2\pi N_{0G}}{A' + B'} (S - 1) * \left[\frac{0.78}{\lambda_G^2} + 0.31 \left(\frac{\bar{a}\rho}{\mu} \right)^{0.5} \left(\frac{p_0}{p} \right)^{0.2} \frac{\Gamma\left(\frac{\bar{b}}{2} + \frac{5}{2} \right)}{\lambda_G^{\frac{\bar{b}}{2}+\frac{5}{2}}} \right] \tag{4.120}$$

Shedding of accreted water: RH84 (A20): Subroutine eqa20g.F

$$PGSHR = PGACR + PGACW \tag{4.121}$$

Enhancement of melting from accretion of rain: RH84 (A21) Subroutine eqa21g.F

$$PGACRM = \frac{-c_w}{L_f} (T - T_0) * PGACR \tag{4.122}$$

Enhancement of melting from accretion of cloud water: RH84 (A22) Subroutine eqa21g.F

$$PGACWM = \frac{-c_w}{L_f} (T - T_0) * PGACW \tag{4.123}$$

Drizzle Parameterization: KK00:

CONVERSION EQUATIONS:

CCN count: (4)

$$\frac{\partial n}{\partial t} = -\frac{\partial u_i n}{\partial x_i} - \left(\frac{\partial N_c}{\partial t} \right)_{activ} + \left(\frac{\partial N_c}{\partial t} \right)_{evap} + \frac{\partial}{\partial x_i} K \frac{\partial n}{\partial x_i} \tag{4.124}$$

Cloud water: (5)

$$\frac{\partial q_c}{\partial t} = -\frac{\partial u_i q_c}{\partial x_i} - \left(\frac{\partial q_c}{\partial t} \right)_{activ} + \left(\frac{\partial q_c}{\partial t} \right)_{evap} - \left(\frac{\partial q_r}{\partial t} \right)_{auto} - \left(\frac{\partial q_r}{\partial t} \right)_{accr} + \frac{\partial}{\partial x_i} K \frac{\partial q_c}{\partial x_i} \tag{4.125}$$

Cloud droplet concentration: (6)

$$\frac{\partial N_c}{\partial t} = -\frac{\partial u_i N_c}{\partial x_i} + \left(\frac{\partial N_c}{\partial t} \right)_{activ} + \left(\frac{\partial N_c}{\partial t} \right)_{evap} - \left(\frac{\partial N_c}{\partial t} \right)_{accr} - \left(\frac{\partial N_c}{\partial t} \right)_{auto} + \frac{\partial}{\partial x_i} K \frac{\partial N_c}{\partial x_i} \tag{4.126}$$

Integral radius of cloud droplets: (7)

$$\frac{\partial R_c}{\partial t} = -\frac{\partial u_i R_c}{\partial x_i} - \left(\frac{\partial R_c}{\partial t} \right)_{activ} + \left(\frac{\partial R_c}{\partial t} \right)_{cond} + \frac{\partial}{\partial x_i} K \frac{\partial R_c}{\partial x_i} \tag{4.127}$$

Drizzle mixing ratio: (8)

$$\frac{\partial q_r}{\partial t} = -\frac{\partial u_i q_r}{\partial x_i} + \frac{\partial V_{q_r} q_r}{\partial z} + \left(\frac{\partial q_r}{\partial t} \right)_{cond} + \left(\frac{\partial q_r}{\partial t} \right)_{auto} + \left(\frac{\partial q_r}{\partial t} \right)_{accr} + \frac{\partial}{\partial x_i} K \frac{\partial q_r}{\partial x_i} \tag{4.128}$$

Drizzle drop concentration: (9)

$$\frac{\partial N_r}{\partial t} = -\frac{\partial u_i N_r}{\partial x_i} + \frac{\partial V_{N_r} N_r}{\partial z} - \left(\frac{\partial N_r}{\partial t}\right)_{evap} + \left(\frac{\partial N_r}{\partial t}\right)_{auto} + \frac{\partial}{\partial x_i} K \frac{\partial N_r}{\partial x_i} \tag{4.129}$$

SOURCE TERMS:

CCN Activation: KK00 (12) Subroutine nrmtqw.F

$$\left(\frac{\partial N_c}{\partial t}\right)_{activ} = \frac{\max\left\{0, (n+N_c)\min\left[1, \left(\frac{S}{S_{\max}}\right)^k\right] - N_c\right\}}{\Delta t} \tag{4.130}$$

Integral radius activation: KK00 (13) Subroutine nrmtqw.F

$$\left(\frac{\partial R_c}{\partial t}\right)_{activ} = r_{act}\left(\frac{\partial N_c}{\partial t}\right)_{activ} \tag{4.131}$$

Condensation/Evaporation: cloud water: KK00 (15,16) Subroutine eqa6.F

$$\left(\frac{\partial q_c}{\partial t}\right)_{cond} = \frac{4\pi G(T,P)\rho_w}{\rho_a} S R_c \tag{4.132}$$

$$\left(\frac{\partial R_c}{\partial t}\right)_{cond} = \frac{G(T,P)}{S N_c}\overline{\left(\frac{1}{r}\right)} \tag{4.133}$$

where: KK00 (19,20,17)

$$\overline{\left(\frac{1}{r}\right)} = \frac{(\gamma+1)N_c}{\gamma R_c} \tag{4.134}$$

and

$$\gamma = \frac{5 - 2P + (8P+1)^{0.5}}{2P-2}; \quad P \equiv \frac{3\rho q_{ca} N_c^2}{4\pi\rho_w R_c^3} \tag{4.135}$$

$$\left(\frac{\partial r}{\partial t}\right)_{cond} = \frac{G(T,P)S}{r} \tag{4.136}$$

Condensation/Evaporation of drizzle: KK00(22) Subroutine eqa12.F

$$\left(\frac{\partial q_r}{\partial t}\right)_{cond} = 3C_{evap}G(T,P)\left(\frac{4\pi\rho_w}{3\rho_a}\right)^{\frac{2}{3}} q_r^{\frac{1}{3}} N_r^{\frac{2}{3}} S \tag{4.137}$$

Drizzle concentration tendency from evaporation: KK00 (23) Subroutine nrmtqw.F

$$\left(\frac{\Delta N_r}{N_r}\right)_{evap} = \left(\frac{\Delta q_r}{q_r}\right)^{\nu}_{evap} \tag{4.138}$$

Autoconversion from cloud water to drizzle: KK00 (29) Subroutine eqa7.F

$$\left(\frac{\partial q_r}{\partial t}\right)_{auto} = 1350 q_c^{2.47} N_c^{-1.79} \tag{4.139}$$

Drizzle concentration tendency from autoconversion: KK00 (32) Subroutine nrmcol.F

$$\left(\frac{\partial N_r}{\partial t}\right)_{auto} = \frac{\left(\frac{\partial q_r}{\partial t}\right)_{auto}}{\left(\frac{4\pi\rho_w}{3\rho_a} r_0^3\right)} \tag{4.140}$$

Accretion of cloud droplets: KK00 (33) Subroutine eqa9.F

$$\left(\frac{\partial q_r}{\partial t}\right)_{accr} = 67(q_c q_r)^{1.15} \tag{4.141}$$

Cloud droplet tendency due to accretion: KK00 (35) Subroutine nrmcol.F

$$\left(\frac{\partial N_c}{\partial t}\right)_{accr} = \frac{\left(\frac{\partial q_r}{\partial t}\right)_{auto} + \left(\frac{\partial q_r}{\partial t}\right)_{accr}}{\left(\frac{4\pi\rho_w}{3\rho_a} r_{vc}^3\right)} \tag{4.142}$$

Sedimentation: KK00 (37) Subroutine tvqr.F, tgqr.F, and tvnr.F

$$V_{N_r} = 0.007 r_{rv} - 0.1 \tag{4.143}$$

$$V_{q_r} = 0.012 r_{rv} - 0.2 \tag{4.144}$$

Table 4.1 — Symbols for Moist Physics Equations

Symbol	Description	Value	SI Units
A'	Thermodynamic term in PREVP		$m\ s\ kg^{-1}$
$\overline{a}$	Thermodynamic term in PDEPI		$m\ s\ kg^{-1}$
a	Constant in fallspeed relation for graupel	19.3	$m^{(1-\ b\ )}\ s^{-1}$
a	Constant in M-Z relation	0.008	$g\ m^{\ 3(b'-1)}\ mm^{\ -6b'}$
a'	Constant in linear fallspeed relation for rain	3×10^3	s^{-1}
a"	Constant in fallspeed relation for snow	1.139	$m^{(1-b)}\ s^{-1}$
a_0	Coefficient in polynomial fallspeed relation for rain	-0.267	$m\ s^{-1}$
a_1	Coefficient in polynomial fallspeed relation for rain	5.15×10^3	s^{-1}
a_2	Coefficient in polynomial fallspeed relation for rain	-1.0225×10^6	$m^{-1}\ s^{-1}$
a_3	Coefficient in polynomial fallspeed relation for rain	7.55×10^7	$m^{-2}\ s^{-1}$
B'	Thermodynamic term		$m\ s\ kg^{-1}$
B"	Thermodynamic term		$m\ s\ kg^{-1}$
$\overline{b}$	Fallspeed exponent of graupel	0.37	
b	Fallspeed exponent for snow	0.11	
b'	Constant in M-Z relation	0.605	
C	Capacitance of ice crystal		F
c_p	Specific heat of air at constant pressure	1.005×10^3	$J\ kg^{-1}\ K^{-1}$
c_w	Specific heat of liquid water at 0° C	4218	$J\ kg^{-1}\ K^{-1}$
D_I	Diameter of hexagonal plate		m
D_0	Initial diameter of cloud ice crystals	12.9×10^{-6}	m
D_G	Graupel diameter		m
D_R	Raindrop diameter		m
D_S	Snowflake diameter		m
$dB(Z_R)$	$10 \log_{10}$ (radar reflectivity factor for rain)		
$dB(Z_S)$	$10 \log_{10}$ (radar reflectivity factor for snow)		
E_{GC}	Graupel/cloud water collection efficiency	1	
E_{GI}	Graupel/cloud ice collection efficiency	0.1	
E_{GR}	Graupel/ rain collection efficiency	1	
E_{GS}	Graupel/snow collection efficiency	0.1	
E_{RC}	Rain/cloud water collection efficiency	1	
E_{RI}	Rain/cloud ice collection efficiency	1	
E_{SC}	Snow/cloud water collection efficiency	1	
E_{SI}	Snow/cloud ice collection efficiency	0.1	
E_{SR}	Snow/rain collection efficiency	1	
$\overline{E_t}$	Average diameter of cloud ice crystals		m
e_{si}	Saturation vapor pressure for ice		$N\ m^{-2}$
e_{sw}	Saturation vapor pressure for water		$N\ m^{-2}$
F	Ventilation factor for rain and graupel		
F'	Ventilation factor for snow		
K_a	Thermal conductivity of air	2.43×10^{-2}	$J\ m^{-1}\ s^{-1}\ K^{-1}$
L_f	Latent heat of fusion	3.34×10^5	$J\ kg^{-1}$
L_s	Latent heat of sublimation	2.5×10^6	$J\ kg^{-1}$
L_c	Latent heat of condensation	2.25×10^6	$J\ kg^{-1}$
$\overline{M_C}$	Average mass of cloud droplet	4×10^{-12}	kg
$\overline{M_I}$	Average mass of cloud ice particle	6×10^{-12}	kg
M_i	Average mass of cloud ice crystal		kg
M_{max}	Maximum mass of cloud ice crystal	9.4×10^{-10}	kg
M_0	Initial mass of cloud ice crystal	10^{-12}	kg
M_G	Mass of graupel per unit volume of air		$kg\ m^{-3}$
M_R	Mass of rain per unit volume of air		$kg\ m^{-3}$
M_S	Mass of snow per unit volume of air		$kg\ m^{-3}$
M_w	Molecular weight of water	18.0160	
M_{melt}	Mass of melted snow		kg

Table 4.1 (continued) — Symbols for Moist Physics Equations

Symbol	Description	Value	SI Units
$M(D_G)$	Mass of graupel particle of diameter D_G		kg
$M(D_R)$	Mass of raindrop of diameter D_R		kg
$M(D_S)$	Mass of snowflake of diameter D_S		kg
$N_{DG}dD_G$	Number of concentration of graupel particles with diameters between D_G and $D_G + d\,D_G$		m^{-3}
$N_{DR}dD_R$	Number of concentration of raindrops with diameters between D_R and $D_R + d\,D_R$		m^{-3}
$N_{DS}dD_S$	Number of concentration of snowflakes with diameters between D_S and $D_S + d\,D_S$		m^{-3}
N_{0G}	Intercept value in graupel size distribution	4×10^6	m^{-4}
N_{0R}	Intercept value in raindrop size distribution	8×10^6	m^{-4}
N_{0S}	Intercept value in snowflake size distribution	4×10^6	m^{-4}
n_{ci}	Number concentration of cloud ice crystals		m^{-3}
$\overline{n}_c$	Number concentration of cloud water droplets		m^{-3}
n_i	Number concentration of ice nuclei		m^{-3}
n_0	Constant in expression for ice nuclei concentration	variable	m^{-3}
p	Pressure		N m^{-2}
p_0	Constant in empirical relation	10^6	N m^{-2}
PCOND	Condensation of water vapor		kg m^{-3} s^{-1}
PCONV	Conversion of cloud ice to snow		kg m^{-3} s^{-1}
PDEPI	Depositional growth of cloud ice		kg m^{-3} s^{-1}
PGACI	Collection of cloud ice by graupel		kg m^{-3} s^{-1}
PGACR	Collection of rain by graupel		kg m^{-3} s^{-1}
PGACRM	Enhanced melting of graupel due to accretion of rain		kg m^{-3} s^{-1}
PGACS	Collection of snow by graupel		kg m^{-3} s^{-1}
PGACW	Collection of cloud water by graupel		kg m^{-3} s^{-1}
PGACWM	Enhanced melting of graupel due to accretion of cloud water		kg m^{-3} s^{-1}
PGDEP	Depositional growth of graupel		kg m^{-3} s^{-1}
PGMLT	Melting of graupel		kg m^{-3} s^{-1}
PGSHR	Shedding of accreted water by graupel		kg m^{-3} s^{-1}
PIACR	Collection of rain by cloud ice		kg m^{-3} s^{-1}
PMLTEV	Evaporation of melting snow		kg m^{-3} s^{-1}
PMLTGE	Evaporation of melting graupel		kg m^{-3} s^{-1}
PRACI	Collection of cloud ice by rain		kg m^{-3} s^{-1}
PRACS	Collection of snow by rain		kg m^{-3} s^{-1}
PRACW	Collection of cloud water by rainwater		kg m^{-3} s^{-1}
PRAUT	Autoconversion of cloud water		kg m^{-3} s^{-1}
PREVP	Evaporation of rainwater		kg m^{-3} s^{-1}
PSACI	Collection of cloud ice by snow		kg m^{-3} s^{-1}
PSACR	Collection of rain by snow		kg m^{-3} s^{-1}
PSACW	Collection of cloud water by snow		kg m^{-3} s^{-1}
PSDEP	Depositional growth of snow		kg m^{-3} s^{-1}
PSFI	Conversion of cloud ice to snow in the Bergeron process		kg m^{-3} s^{-1}
PSFW	Conversion of cloud water to snow in the Bergeron process		kg m^{-3} s^{-1}
PSMLT	Melting of snow		kg m^{-3} s^{-1}
PSMLTI	Melting of cloud ice		kg m^{-3} s^{-1}
PINT	Initiation of cloud ice		kg m^{-3} s^{-1}
q_c	Mixing ratio of cloud water		kg kg^{-1}
q_i	Mixing ratio of cloud ice		kg kg^{-1}
q_{imax}	Conversion of cloud ice to snow threshold		kg kg^{-1}
q_0	Mixing ratio threshold for PRAUT	7×10^{-4}	kg kg^{-1}
q_r	Mixing ratio of rainwater		kg kg^{-1}
q_s	Mixing ratio of snow		kg kg^{-1}

Table 4.1 (continued) — Symbols for Moist Physics Equations

Symbol	Description	Value	SI Units
q_{s0}	Mixing ratio of snow at top of feeder zone		kg kg^{-1}
q_{si}	Saturation mixing ratio with respect to ice		kg kg^{-1}
q_{sw}	Saturation mixing ratio with respect to water		kg kg^{-1}
q_v	Mixing ratio of water vapor		kg kg^{-1}
R^*	Universal gas constant	8.314×10^3	J kmol^{-1} K^{-1}
R_e	Reynolds number		
R_w	Gas constant for water vapor	4.61×10^2	J kg^{-1} K^{-1}
S	Saturation ratio with respect to water		
S_c	Schmidt number	0.6	
S_C	Source term for cloud water		kg m^{-3} s^{-1}
S_h	Diabatic heating terms		K kg m^{-3} s^{-1}
S_I	Source term for cloud ice		kg m^{-3} s^{-1}
S_i	Saturation ratio with respect to ice		kg m^{-3} s^{-1}
S_0	Represents sources and sinks for q		kg m^{-3} s^{-1}
S_R	Source term for rain		kg m^{-3} s^{-1}
S_S	Source term for snow		kg m^{-3} s^{-1}
S_V	Source term for water vapor		kg m^{-3} s^{-1}
T	Temperature		K
T_0	Reference temperature	273.16	K
t	Time		s
u	Horizontal windspeed		m s^{-1}
V	Mass-weighted fallspeed of precipitation		m s^{-1}
V_R	Mass-weighted fallspeed for rain		m s^{-1}
$V_R(D_R)$	Fallspeed of raindrop of diameter D_R		m s^{-1}
V_S	Mass-weighted fallspeed for snow		m s^{-1}
$V_S(D_S)$	Fallspeed of snowflake for diameter D_S		m s^{-1}
w	Vertical air velocity		m s^{-1}
x	Horizontal distance		m
Z	Equivalent radar reflectivity factor		mm^6 m^{-3}
z	Vertical distance		m
α	Rate coefficient for autoconversion	0.001	s^{-1}
β	Constant in ice crystal concentration	0.6	deg^{-1}
Γ	Gamma function		
Γ_d	Dry adiabatic lapse rate	9.8×10^{-3}	K m^{-1}
ε_0	Permittivity of free space	8.854×10^{-12}	C^2 N^{-1} m^{-2}
ρ	Air density		kg m^{-3}
ρ_L	Density of water	10^3	kg m^{-3}
ρ_S	Density of snow	100 (Type 1) 200 (Type 2)	kg m^{-3} kg m^{-3}
λ_R	Slope of raindrop size distribution		m^{-1}
λ_S	Slope of snow size distribution		m^{-1}
χ	Diffusivity of water vapor in air	2.26×10^{-5}	m^2 s^{-1}
μ	Dynamic viscosity of air	1.718×10^{-5}	kg m^{-1} s^{-1}
Δ_t	Time increment	10	s
Δ_x	Horizontal spatial increment	4000	m
Δ_z	Vertical spatial increment	200	m

4.3 Cumulus Scheme

Convective clouds can produce a significant amount of mixing. This is not resolved on the COAMPS model grid when horizontal grid dimensions are greater than 4 km (Weisman et al. 1997). The convective parameterization of Kain and Fritsch (1990, 1993) accounts for the mixing and attendant precipitation. The Kain-Fritsch scheme is fundamentally similar to the scheme of Fritsch-Chappell (1980), but with two primary improvements. The most important of these improvements is the computation of detrainment effects. This involves using a new cloud model that determines the profile of mixing of updraft air with the environment through the implementa-

tion of the Raymond and Blyth (1986) stochastic buoyancy-sorting model. While this improvement is indispensable, it has an expense of added computational time. The second notable improvement to the Kain-Fritsch scheme is the formulation to assure conservation of mass, thermal energy, total moisture, and momentum. The Kain-Fritsch scheme has undergone some modifications since it was described in the literature, but the fundamental structure of the scheme remains unchanged.

The convective scheme is applied to COAMPS when the *coamnl* namelist variable *icup* = 3 (default) (Figure 8). Because of the uncertainty with parameterization of convection for finer scales, the convective scheme is currently used only for horizontal grid dimensions greater than 9 km. This threshold is controlled by the *coamnl* namelist variable *dxmeso*.

4.3.1 General framework

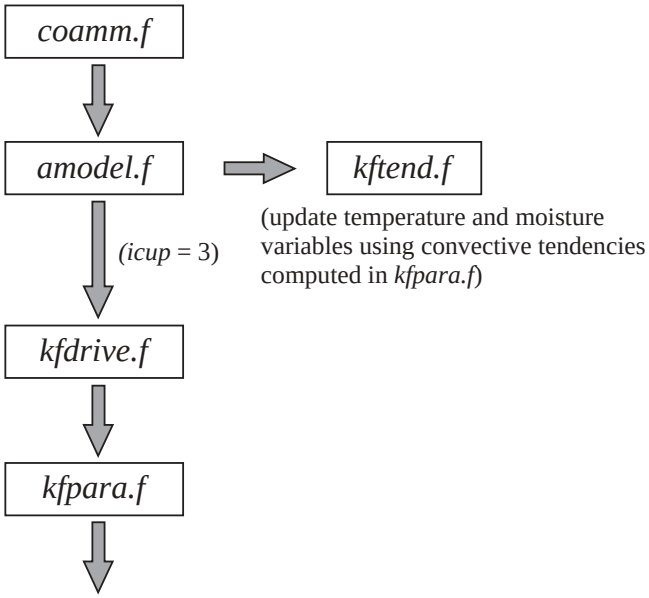


Figure 8
Flow chart of the COAMPS convective scheme with the default setting of variable icup from the *coamnl* namelist (where icup = 3).

The Kain-Fritsch scheme computes tendencies of potential temperature, water vapor mixing ratio, and cloud water due to convection between a selected source layer and the corresponding cloud top. These tendencies are computed in the main convective parameterization routine *kfpara.f* at the initiation of convection in a given grid column. These values remain constant through the assumed duration *timec* of the convection, which can vary between 1800 and 3600 seconds. The corresponding grid-scale quantities are updated at each time step during the convective period in the routine *kftend.f*. The convective rainfall is also computed in *kfpara.f*.

The Kain-Fritsch scheme is initiated from the routine *amodel.f* by calling *kfdrive.f*. This routine performs a rough screening for selecting grid points that is consistent with the occurrence of convection. The main convective subroutine *kfpara.f* is called from *kfdrive.f*.

The tendencies of potential temperature q and water vapor mixing ratio q_v are computed by:

$$\left.\frac{\partial \theta}{\partial t}\right|_{conv} = \frac{L}{\pi}\overline{\frac{dq}{dt}} - \frac{\partial\left(\overline{\omega'\theta'}\right)}{\partial p}, \tag{4.145}$$

$$\left.\frac{\partial q_v}{\partial t}\right|_{conv} = \overline{\frac{dq_v}{dt}} - \frac{\partial\left(\overline{\omega' q_v'}\right)}{\partial p}. \tag{4.146}$$

The L term represents latent heat released from changes in q_v due to change of phase. The p term represents pressure, π is Exner's function, and ω is dp/dt. Overbars indicate grid-scale mean values, and primes denote subgrid-scale variations. The second term on the right-hand side of these equations is approximated by the sum of contributions associated with mean values of w within the updraft, downdraft, and environment due to compensating subsidence. Local (grid-scale) compensation of the corresponding contributions to is assumed with:

$$\overline{\omega} \approx \omega_u + \omega_d + \tilde{\omega} = 0. \tag{4.147}$$

The terms w_u, w_d, and represent contributions to from the updraft, downdraft and environmental subsidence, respectively. The discretized versions of the resulting equations are

$$\left.\frac{\Delta\overline{\theta}}{\Delta t}\right|_{conv} = \frac{1}{\Delta p}\left[\left(\omega_{u2}+\omega_{d2}\right)\overline{\theta}_2 - \left(\omega_{u1}+\omega_{d1}\right)\overline{\theta}_1 + \left(\varepsilon_u+\varepsilon_d\right)\overline{\theta}_m - \delta_u\theta_{um} - \delta_d\theta_{dm}\right], \tag{4.148}$$

$$\left.\frac{\Delta\overline{q_v}}{\Delta t}\right|_{conv} = \frac{1}{\Delta p}\left[\left(\omega_{u2}+\omega_{d2}\right)\overline{q}_{v2} - \left(\omega_{u1}+\omega_{d1}\right)\overline{q}_{v1} + \left(\varepsilon_u+\varepsilon_d\right)\overline{q}_{vm} - \delta_u q_{vum} - \delta_d q_{vdm}\right]. \tag{4.149}$$

These equations are solved in *kfpara.f* using an upstream, forward-in-time advection scheme. In these equations, subscripts "1," "2," and "*m*" denote bottom, top, and mean values, respectively, for a given model level. Subscripts "*u*" and "*d*" denote updraft and downdraft values. The rates of entrainment of environmental mass and detrainment of either updraft or downdraft mass are represented by ε and δ, respectively, with appropriate subscripts.

The version of the Kain-Fritsch scheme in COAMPS feeds back to the grid-scale condensate detrained from updrafts in the form of cloud water by using:

$$\left.\frac{\Delta\overline{q_c}}{\Delta t}\right|_{conv} = -\frac{\delta_u q_{lum}}{\Delta p}. \tag{4.150}$$

The term q_c represents cloud water, and q_{lum} is the total amount of condensate in the updraft. For deep convection, this term is generally very small. Most of the condensate either falls to the surface as convective rainfall or is evaporated in the downdraft. Although Kain and Fritsch (1993) include a treatment of convective momentum transport, assuming conservation of momentum in convective updrafts, it is not provided in the code currently used in COAMPS.

4.3.2 Cloud model

The equations in Section 4.3.1 reduce the convective parameterization problem for determining the vertical profiles of ω_u, ω_d, ε_u, ε_d, δ_u, and δ_d, as well as the vertical profiles of θ and q_v in the updraft and downdraft, and the liquid and ice condensate profiles in the updraft. The solution provided by the Kain-Fritsch scheme is based on a new entraining-detraining plume model of convective updrafts. The amount of environmental air that mixes with the updraft at each level is inversely proportional to the assumed updraft radius, set by variable *rad* in *kfpara.f* (by default, *rad* = 1500 m). The buoyancy-sorting treatment establishes the amount of entrainment and detrainment between the updraft and the environment at each level. The entrainment and detrainment factor in the buoyancy of individual subparcel mixtures assuming mixing distribution between cloudy air and environmental air. In general, greater updraft buoyancy and/or environmental relative humidity result in an increasing updraft mass flux, with height due to net entrainment of air into the updraft.

The removal of condensate from the updraft as precipitation is parameterized following the methodology of Ogura and Cho (1973). The change in total condensate δr_c in a layer of thickness δz due to precipitation is computed by

$$\delta r_c = r_{c0}\left(1 - e^{-c_1 \delta z / w}\right). \tag{4.151}$$

The r_{c0} term is the sum of condensate concentration at the bottom of the layer and half the degree of supersaturation at the top. The w term represents the layer-mean vertical velocity, and c_1 is a rate constant. A linear transition is assumed from θ_e with respect to liquid water to θ_e with respect to ice for temperatures, ranging from 268 to 248 K. The conversion of liquid condensate to ice is modeled in a manner consistent with the assumed profile of θ_e.

The Kain-Fritsch scheme includes a downdraft (Kain and Fritsch 1993) component that is modeled in *kfpara.f* after the updraft properties have been determined. The downdraft calculation removes mass flux and condensate from the updraft. Condensate evaporates in the downdraft while maintaining a specified relative humidity (100 percent above cloud base and 90 percent below cloud base). The downdraft mass flux obtains agreement with the parameterized precipitation efficiency. This is calculated as the mean of estimated vertical shear of the horizontal wind (Fritsch and Chappel 1980) and estimated cloud base height (Fujita 1959; Zhang and Fritsch 1986).

4.3.3 Closure treatment

The cloud model determines the vertical profiles of the vertical mass flux as well as the amount of entrained and detrained mass flux for both updraft and downdraft. The quantities ω_u, ω_d, ε_u, ε_d, δ_u, and δ_d, are obtained from the corresponding mass fluxes. Equation (4.152) is an example of ω_u:

$$\omega_u = -M_u g / A. \tag{4.152}$$

The term M_u represents the updraft mass flux, g is the acceleration due to gravity, and A is the horizontal area of a model grid cell. Although the cloud model establishes the vertical profiles of the various mass fluxes, the magnitudes are determined by the closure scheme in *kfpara.f*. The computation is performed through an iterative procedure that attempts to satisfy the condition that 90 percent of the initial available buoyant energy (Fritsch and Chappell 1980) is removed during the convective time-scale (*timec*).

4.3.4 Standard printouts

Several diagnostic printouts are currently used in the main Kain-Fritsch routine *kfpara.f* to report problems. Table 4.2.1 summarizes these statements and their significance.

Table 4.2 — Significance of Diagnostic Printouts from *kfpara.f*

Diagnostic Printout	Significance
in *kfpara*, l5 out of range, l5 =”integer”	correction to convective time scale timec, which accounts for advection, is invalid – run continues
!!!!! problem with kf scheme: qg = 0 at the surface!!!!!!! stop qg	water vapor mixing ratio is negative at the first model level – run aborts
!!!warning!!! cloud base water vapor changes by ___ % when moisture is borrowed to prevent negative values in kain-fritsch	unrealistic drying of certain layers due to parameterized convection – run continues
stop kain-fritsch	error in the mass budget – run aborts

4.4 COAMPS Radiation Module

4.4.1 Theory and Equations

The radiative transfer parameterization in COAMPS was inherited from its predecessor, the Navy Operational Regional Atmospheric Prediction System (NORAPS). NORAPS codes were adapted from the radiative transfer parameterization used in the Navy Operational Global Atmospheric Prediction System (NOGAPS, Hogan and Rosmond 1991). The parameterization includes both shortwave and longwave radiative transfer calculations following the methods of Harshvardhan et al. (1987).

Two types of clouds are considered in the radiative transfer calculation: stratiform and cumulus. The fractional coverage of stratiform clouds is diagnosed by different critical values of relative humidity at each model level. The fractional coverage of cumulus clouds is diagnosed by convective rainfall rate and temperature. The total cloud fraction is obtained by assuming that the two types of clouds are randomly overlapped.

4.4.1.1 Longwave Radiative Transfer

The longwave calculation considers longwave radiation absorption by water vapor, carbon dioxide, and ozone. The absorption is computed by a broadband method of four band regions: (1) the water vapor band center (5.26-7.25 μm, 29.41-∞ μm), (2) the water vapor band wing (3.30-5.26 μm, 7.25-9.09 μm, 10.20-12.50 μm, 18.52-29.41 μm), (3) the 15 μm carbon dioxide band (12.50-18.52 μm), and (4) the 9 μm ozone band (9.09-10.20 μm). Clouds are considered blackbodies in the longwave radiative transfer. The effects of clouds on longwave fluxes are incorporated by multiplying clear sky fluxes with the probability of a clear line of sight between model levels. Specifically, the temperature change due to longwave radiation is calculated by

$$C_p \rho \frac{\partial T}{\partial t} = -\frac{\partial}{\partial z}(F^{\uparrow} - F^{\downarrow}). \tag{4.153}$$

The term $F^{\uparrow}$ is the upward longwave flux and $F^{\downarrow}$ is the downward longwave flux. The longwave flux for a given frequency is computed by Plank flux from the temperature by

$$B_{\nu} = \frac{2h\nu^3}{c^2(e^{h\nu/KT} - 1)}. \tag{4.154}$$

The term ν is the frequency, h is Planck's constant (6.6262×10^{-27} erg sec), K is Boltzmann's constant (1.3806×10^{-16} erg/K), and c is the speed of light. The total upward and downward longwave fluxes for the four broad bands is computed by applying the diffuse transmission function Γ and considering cloud effects through the probability of a clear line of sight P with the following equations:

$$F_k^{\uparrow} = \sum_{i=1}^{4}\{B_{ik} + (B_{ig} - B_{is}) \cdot \Gamma_s^k \cdot P_s^k\} - \sum_{i=1}^{4}\sum_{j=1}^{N-k}\{(B_{i(k+j-1)} - B_{i(k+j)}) \cdot \Gamma_{(k+j)}^k \cdot P_{(k+j)}^k\}, \tag{4.155}$$

$$F_k^{\downarrow} = \sum_{i=1}^{4}\{B_{ik} - B_{iptop} \cdot \Gamma_{ptop}^k \cdot P_{ptop}^k\} + \sum_{i=1}^{4}\sum_{j=1}^{N-k}\{(B_{i(k-j-1)} - B_{i(k-j)}) \cdot \Gamma_{(k-j)}^k \cdot P_{(k-j)}^k\}. \tag{4.156}$$

The term k is the index for vertical level, g is the index for earth ground, s is the index for air surface, $ptop$ is the index for model top, N is the total number of vertical levels, Γ_b^a is the diffuse transmission between levels a and b, and P_a^b is the probability of a clear line of sight between levels a and b. The probability of a clear line of sight is computed by assuming the maximum overlap of the cloud fraction in all different levels. The term B_{ik} is defined as:

$$B_{ik} = \int_{(\Delta\nu)i} \pi B_{\nu k} d\nu. \tag{4.157}$$

4.4.1.2 Shortwave radiative transfer

The shortwave calculation considers absorption by water vapor and ozone. The multiple scattering of shortwave radiation in cloudy and clear skies is determined by using the adding method for two-stream solutions (Davies 1982). The adding method combines direct and diffuse fluxes with a magnification factor for multiple reflections. The upward and downward fluxes are computed by using the two-stream approximation of the delta-Eddington method (Joseph et al. 1976). The diffuse reflection and transmission coefficients are computed from the Sagan and Pollack (1967) quadrature two-stream solution for diffuse radiation. Specifically, the temperature change due to shortwave radiation is calculated by

$$C_p \rho \frac{\partial T}{\partial t} = -\frac{\partial A}{\partial z}. \tag{4.158}$$

The term A is the absorbed shortwave flux. It is assumed that the absorption of ozone takes place only for wavelengths shorter than 0.9 μm and at levels above cloud tops or above 500 mb. Following the method of Lacis and Hansen (1974), the ozone absorption in the visible frequencies is parameterized with

$$A_{oz-v}(x) = \frac{0.02118x}{1 + 0.042x + 0.000323x^2}. \tag{4.159}$$

The absorption in the ultraviolet frequencies is parameterized by

$$A_{oz-u}(x) = \frac{1.082x}{(1+136.6x)^{0.805}} + \frac{0.0658x}{1+(103.6x)^3}. \tag{4.160}$$

The x term is the effective path computed by the multiplication of ozone optical depth τ and a magnification factor M. The total absorbed shortwave flux by the ozone at level k is computed using

$$A_{ozk} = S_0 \cos\theta_0 \{(A_{oz-v}(x) + A_{oz-u}(x))_k + R_{oz}(A_{oz-v}(x^*) + A_{oz-u}(x^*))_k\}. \tag{4.161}$$

The S_0 term is the solar constant, θ_0 is the solar angle, R_{oz} is the total albedo of the underlying atmosphere to visible and ultraviolet light, and x^* is the reflected radiation effective path.

It is assumed that absorption of water vapor occurs only for wavelengths longer than 0.9 μm. Reflection R and transmission T coefficients for shortwave scattering by clouds are computed by five different absorption coefficients k, with five associated probability distributions p as $T(x) = \sum_{i=1}^{5} p(k_i)e^{-k_i x}$, where $k_1 = 0.005$, $k_2 = 0.041$, $k_3 = 0.416$, $k_4 = 4.752$, $k_5 = 72.459$, $p(k_1) = 0.107$, $p(k_2) = 0.104$, $p(k_3) = 0.073$, $p(k_4) = 0.044$, and $p(k_5) = 0.025$. The adding method provides the two-stream solutions with absorbed shortwave flux at level k using

$$A_{wvk} = D_k - U_k = (\hat{D}_k + \hat{R}_k U_k + S_k) - (\hat{U}_k + \hat{M}_k \hat{\Gamma}_k U_{(k+1)}). \tag{4.162}$$

The D_k term is the total downward flux, U_k is the total upward flux, $\hat{D}_k$ is the downward flux of the adding solution calculated by

$$\hat{D}_k = \tilde{D}_k + \hat{M}_k t_k (\hat{R}_k \tilde{U}_k + \hat{D}_{(k-1)}). \tag{4.163}$$

The $\hat{U}_k$ term is the upward flux of the adding solution calculated by

$$\hat{U}_k = \hat{M}_k (\tilde{U}_k + \gamma_k \hat{D}_k). \tag{4.164}$$

The $\hat{R}_k$ term is the composite reflection coefficient, S_k is the direct solar flux, $\hat{M}_k$ is the magnification factor, $\hat{\Gamma}_k$ is the total reflection coefficient, $\tilde{D}_k$ is the downward diffuse flux of the two-stream solution, $\tilde{U}_k$ is the upward

diffuse flux of the two-stream solution, t_k and γ_k are diffuse transmission and reflection coefficients, respectively. The diffuse transmission and reflection coefficients are calculated by

$$t_k = \frac{4u_k}{(u_k+1)^2 e^{z_k} - (u_k-1)^2 e^{-z_k}}, \tag{4.165}$$

$$\gamma_k = \frac{(u_k^2-1)(e^{z_k} - e^{-z_k})}{(u_k+1)^2 e^{z_k} - (u_k-1)^2 e^{-z_k}}. \tag{4.166}$$

The u_k and z_k terms for the diffuse transmission and reflection coefficients are calculated using g = 0.85 in the following:

$$u_k = \sqrt{\left(\frac{1-g\omega_{0k}}{1-\omega_{0k}}\right)}, \tag{4.167}$$

$$z_k = \tau_k^* \sqrt{3(1-\omega_{0k})(1-g\omega_{0k})}. \tag{4.168}$$

The ω_{0k} term is single scattering albedo, and τ_k^* is total optical depth. The two-stream solution of the downward and upward diffuse fluxes are calculated by

$$\tilde{D}_k = S_k \hat{t}_k, \tag{4.169}$$

$$\tilde{U}_k = S_k \cdot \hat{\gamma}_k. \tag{4.170}$$

The transmission and reflection coefficients for direct beams are computed by the Delta-Eddington approximation with the following equations:

$$\hat{t}_k = \frac{-\omega_{0k}(1+\kappa\mu_0)(\alpha_1+\kappa\chi_{k4})e^{\kappa\tau_k'} - (1-\kappa\mu_0)(\alpha_1-\kappa\chi_{k4})e^{-\kappa\tau_k'} - 2\kappa(\chi_{k4}+\alpha_1\mu_0)e^{-\tau_k'/\mu_0}}{(1-\kappa^2\mu_0^2)[(\kappa+\chi_{k1})e^{\kappa\tau_k'} + (\kappa-\chi_{k1})e^{-\kappa\tau_k'}]}, \tag{4.171}$$

$$\hat{\gamma}_k = \frac{\omega_{0k}(1-\kappa\mu_0)(\alpha_2+\kappa\chi_{k3})e^{\kappa\tau_k'} - (1+\kappa\mu_0)(\alpha_2-\kappa\chi_{k3})e^{-\kappa\tau_k'} - 2\kappa(\chi_{k3}-\alpha_2\mu_0)e^{-\tau_k'/\mu_0}}{(1-\kappa^2\mu_0^2)[(\kappa+\chi_{k1})e^{\kappa\tau_k'} + (\kappa-\chi_{k1})e^{-\kappa\tau_k'}]}. \tag{4.172}$$

The κ term is the equivalent absorption coefficient, μ_0 is sine of the solar angle, and $\tau_k' = \tau_k^* (1–g^2\omega_{0k})$. The Delta-Eddington parameter χ_k is defined as

$$\chi_1 = \frac{1}{4[7-\omega_0'(4+3g')]}, \tag{4.173}$$

$$\chi_2 = \frac{1}{4[1-\omega_0'(4-3g')]}, \tag{4.174}$$

$$\chi_3 = \frac{1}{4(2-3\mu_0 g')}, \tag{4.175}$$

$$\chi_4 = 1-\chi_3, \tag{4.176}$$

$$\alpha_1 = \chi_1\chi_4 - \chi_2\chi_3, \tag{4.177}$$

$$\alpha_2 = \chi_1\chi_3 + \chi_2\chi_4, \tag{4.178}$$

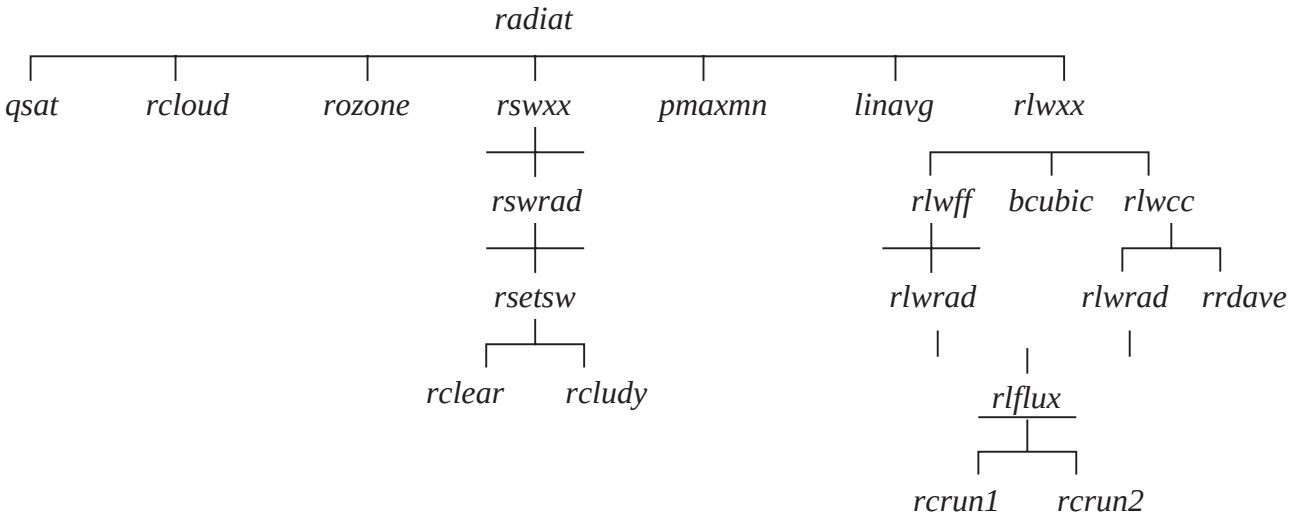
$$\kappa = \sqrt{(\chi_1^2 - \chi_2^2)}, \tag{4.179}$$

$$g' = \frac{g}{(1+g)}, \tag{4.180}$$

$$\omega_0' = \frac{(1-g^2)\omega_0}{(1-g^2\omega_0)}. \tag{4.181}$$

All coefficients and fluxes are computed by summing the results of the five band regions.

4.4.2 Flow chart



4.4.3 coamnl namelist

Table 4.3 — *coamnl* Variables Required by the COAMPS Radiation Module

Name	Type	Description	Default Value
lrad	Logical	Turn on the radiation calculation. = t, turn on both longwave and shortwave calculation = f, turn off the radiation calculation	t
dtrad	Real	Time interval in seconds to update radiative heating rate by calling “radiat”.	3600
njump	Integer	Grid interval in longwave calculation.	1
ldgrad	Logical	Print diagnostic output of longwave and shortwave fluxes at a selected point (*idgrad, jdgrad*). = t, print diagnostics = f, do not print	false
idgrad	Integer	I-coordinate of the selected point for *ldgrad.*	2
jdgrad	Integer	J-coordinate of the selected point for *ldgrad.*	2

4.4.4 Radiation output

```
asl-kk:   max=  0.88975E+01 at (-99,-99)  min=  0.92135E-01 at (-99,-99)
atl-kk:   max=  0.20488E+02 at (-99,-99)  min= -0.18645E+03 at (-99,-99)
mean radiative short/longwave fluxes at sfc (watt/m2)
and htg rate (c/day) at each lvl:
624.7 101.2;
   6.9     1.6     0.8     0.2     0.0    -0.1    -0.1    -0.1    -0.2    -0.4
  -0.4    -0.5    -0.4    -0.4    -0.2    -0.2    -0.2    -0.1    -0.1    -0.1
  -0.1    -0.6    -0.6    -0.7    -0.5    -0.1     0.4     0.9     2.3     3.2
```

The output is interpreted as follows:

- asl-kk = heating rate (k/day) due to shortwave radiative transfer,
- atl-kk = heating rate (k/day) due to longwave radiative transfer,
- 624.7 = domain mean downward shortwave flux (watt/m2) at surface,
- 101.2 = domain mean downward longwave flux (watt/m2) at surface,
- 6.9, 1.6… 2.3, 3.2 = domain mean heating rate (k/day) due to total radiative transfer at each model level.

4.5 Surface and Planetary Boundary Layer Parameterization

4.5.1 COAMPS boundary layer/turbulence

COAMPS uses a level 2.5 scheme (Mellor and Yamada 1982) that solves both a prognostic equation for turbulent kinetic energy (TKE) and diagnostic equations for second-moment quantities such as primarily fluxes of heat, moisture, and momentum. Schemes that feature a prognostic equation for TKE and diagnostic equations for other quantities are referred to as "1.5 order closure." Thus, COAMPS contains a 1.5 order closure, level 2.5 scheme.

The 1.5 order closure, level 2.5 scheme computes all of the fields necessary for solving the TKE equation, including boundary layer depth, turbulent mixing length, flux, Richardson number, and eddy coefficients. In addition, the scheme incorporates the influence of boundary layer cloudiness.

4.5.1.1 Turbulent kinetic energy equation

TKE is calculated using

$$\frac{D}{Dt}(e)-\frac{\partial}{\partial z}\left(K_e\frac{\partial}{\partial z}(e)\right)=K_M\left(\frac{\partial U}{\partial z}\right)^2+K_M\left(\frac{\partial V}{\partial z}\right)^2-\beta g K_H\frac{\partial \theta}{\partial z}-\frac{(2e)^{3/2}}{\Lambda_1}+U\frac{\partial}{\partial x}(e)^*+V\frac{\partial}{\partial y}(e)^*. \quad (4.182)$$

where U and V denote the mean horizontal velocity field, β is the coefficient of thermal expansion, θ is potential temperature, g is the acceleration due to gravity, Λ_1 is the dissipation length scale, u', v', and w' denote the components of the three-dimensional turbulent velocity field, and $K_{H,M,e}$ are eddy coefficients.

The first term on the left side of the TKE equation is diffusion, and the second term is turbulent diffusion of TKE. On the right side of the equation:

- the first and second terms are shear (or mechanical) production of TKE;
- the third term is buoyant production of TKE;
- the fourth term is the dissipation term equal to TKE raised to the 3/2 power divided by the dissipation length scale; and
- the fifth and sixth terms are advective terms used only when the grid increment is less than 10 km.

The prognostic variable is

$$e = \overline{(u'^s + v'^s + w'^s)} / 2. \tag{4.183}$$

In the TKE expression,

$$K_{H,M,e} = S_{H,M,e} l \sqrt{2e}\,. \tag{4.184}$$

where $S_{H,M}$ are polynomial functions of the flux Richardson number, S_e is a constant, and l is the master length scale.

The TKE equation is solved explicitly, omitting the diffusion term. The diffusion term is then implicity solved for later in the code. The solution uses a tri-diagonal technique (a method to solve a set of linear algebraic equations) to invert the coefficient matrix.

4.5.1.2 Effect of clouds

Depending on stability, the buoyancy term in the TKE equation is either a source or a sink of turbulence. In clear conditions, neutral stratification implies that the lapse rate is adiabatic and the virtual potential temperature is conserved. In a cloudy atmosphere, neutral stratification implies that the lapse rate is moist, adiabatic, and virtual potential temperature is not conserved.

The solution to this problem in COAMPS is to use two different temperature variables: virtual potential temperature in clear air and equivalent potential temperature in clouds/fog. The cloud/no-cloud determination is based on *cond*, which is heating/cooling due to condensation/evaporation.

If *cond* is less than or equal to zero at a particular grid point,
- no cloud and/or evaporation is occurring, and
- the vertical temperature gradient is based on virtual potential temperature.

If *cond* is greater than zero at a particular grid point,
- cloud/fog and/or condensation is occurring, and
- the vertical temperature gradient is based on liquid water potential temperature.

4.5.1.3 Boundary layer depth

The boundary layer depth based on the Richardson number (*Ri*), which is the ratio of the buoyant production of TKE to the shear production (see Section 4.5.1.5). The boundary layer depth is the lowest level at which *Ri* exceeds a critical value equal to 0.5. Theoretically, turbulence should occur for *Ri* less than 0.25 for a continuously stratified fluid. When the equations are cast in a finite form, this critical value becomes less certain.

4.5.1.4 Turbulent mixing length scale formulation

There may be as many turbulent mixing length scale formulations as there are models. Many formulations use a function of boundary layer depth, which has undesirable consequences. Following the methods of Blackadar (1962), COAMPS uses

$$l = \frac{\kappa z}{1 + \frac{\kappa z}{\lambda}}, \tag{4.185}$$

where

$$\lambda = \alpha \frac{\int z e dz}{\int e dz}, \tag{4.186}$$

and where κ is the von Karaman constant and z is elevation.

This formulation is commonly used with α set to a constant value. However, in COAMPS the α term is dependent on stability. That is, for z/L greater than zero (stable stratification) α is equal to 0.1, and for z/L less than zero (unstable stratification), α is given by

$$\alpha = 0.1 - \frac{0.2}{3} z/L. \tag{4.187}$$

Figure 9 is an example of the vertical profile of mixing length.

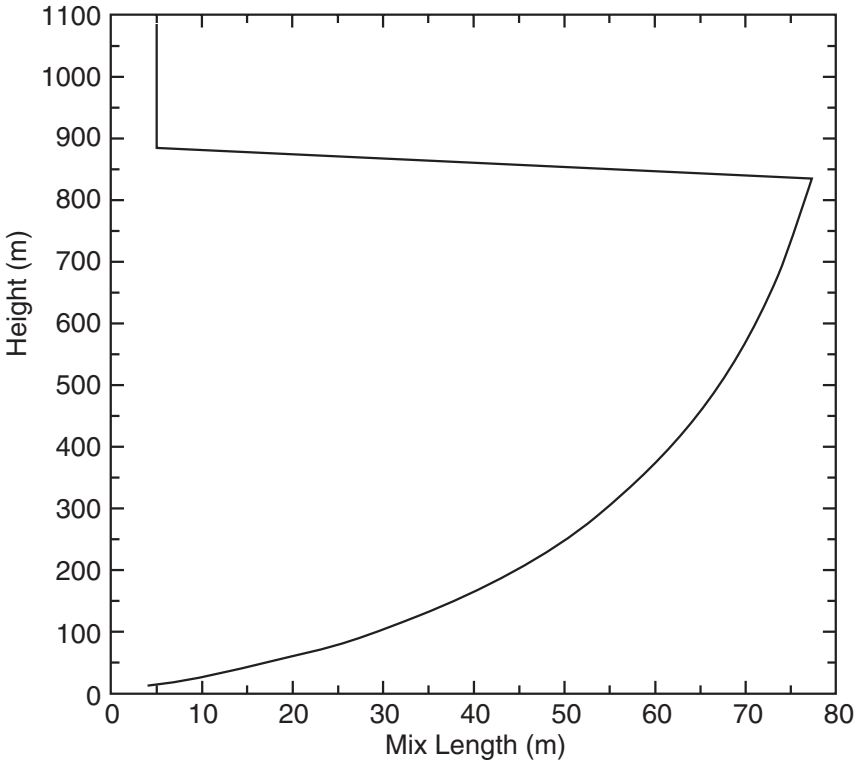


Figure 9.
Mixing length for a nocturnal stratus simulation

4.5.1.5 Eddy coefficient formulation

Early K-theory models used a constant or a prescribed value (cubic polynomial). Most current models use functions of some variation on the Richardson number. COAMPS uses the flux Richardson number following Mellor and Yamada (1982).

The flux Richardson number is the ratio of buoyant production of TKE to shear production of TKE using

$$Ri_f = \frac{g/\theta_v \overline{w'\theta_v'}}{\overline{u'w'}\frac{\partial U}{\partial z} + \overline{v'w'}\frac{\partial V}{\partial z}}, \tag{4.188}$$

where θ is the virtual potential temperature. Polynomial functions of Ri_f, S_H, and S_M are then used to compute the eddy coefficients

$$K_{H,M} = S_{H,M} l\sqrt{2e}. \tag{4.189}$$

Figure 10 shows examples of the vertical profiles of K_M and K_H.

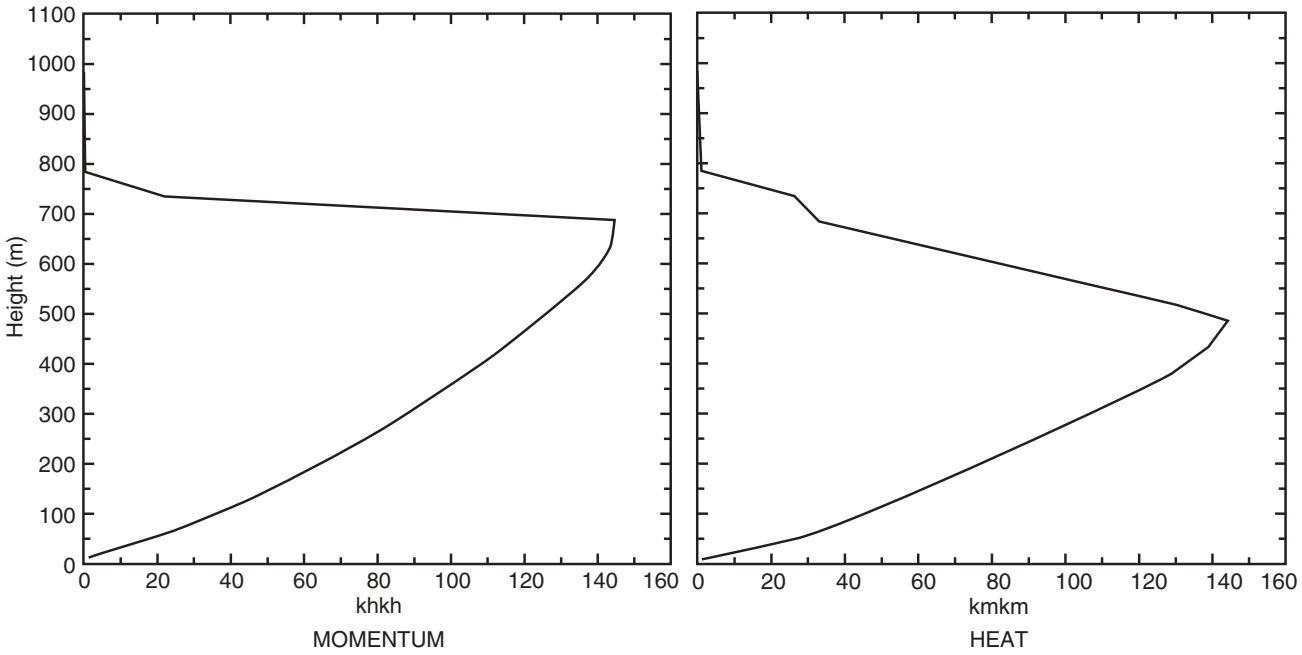


Figure 10.
Eddy coefficients for nocturnal stratus simulation

The order of operations performed in the COAMPS turbulence scheme is listed below.

1. Calculate shear (Section 4.5.1.1)
2. Calculate temperature gradient (Section 4.5.1.2)
3. Calculate buoyancy (Section 4.5.1.2)
4. Calculate boundary layer depth (Section 4.5.1.3)
5. Calculate mixing length (Section 4.5.1.4)
6. Calculate Richardson number (Section 4.5.1.5)
7. Calculate S_M and S_H (Section 4.5.1.5)
8. Add buoyancy and shear production
9. Compute K_M and K_H for next time level (Section 4.5.1.5)
10. Update TKE.

Figure 11 is an example of the vertical profile of TKE.

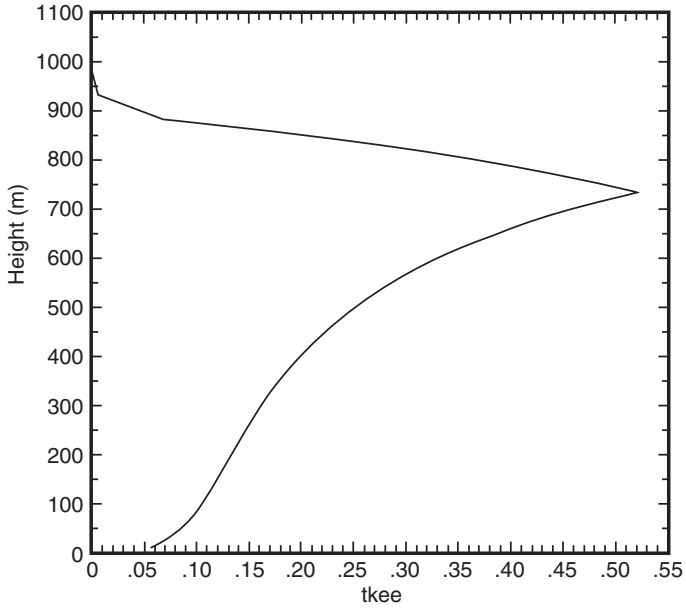


Figure 11.
Forecast turbulent kinetic energy profile

4.5.2 COAMPS surface layer parameterization

4.5.2.1 Louis scheme

The surface layer parameterization follows the Louis (1979) scheme, which uses polynomial functions (F_M, F_H) of the bulk Richardson number to directly compute surface sensible heat flux, surface latent heat flux, and surface drag. The bulk Richardson number is defined as

$$Ri_B = \frac{gz\Delta\theta}{u^2\Theta}, \tag{4.190}$$

where g is the acceleration due to gravity, z is the reference elevation (equal to 10 m in COAMPS), $\Delta\theta$ is the air-sea temperature difference, u is the wind speed at the reference elevation, and Θ is the mean potential temperature over the depth of the surface layer.

The surface fluxes are obtained directly from the polynomials F_M and F_H, with similar expressions for the latent heat flux using

$$u_*^2 = a^2 u^2 F_M\left(\frac{z}{z_0}, Ri_B\right) \tag{4.191}$$

and

$$u_*\theta_* = \frac{a^2}{R} u\Delta\theta F_H\left(\frac{z}{z_0}, Ri_B\right), \tag{4.192}$$

where

$$a^2 = \frac{\kappa^2}{(\ln(z/z_0))^2}. \tag{4.193}$$

The a^2 term is the neutral drag coefficient, z_0 is the surface roughness, κ is the von Karaman constant, and R is the ratio of the transfer coefficient for heat to that for momentum (0.74).

4.5.2.2 Surface roughness

Surface roughness (Fairall et al. 1996) is obtained by

$$z_0 = c_0 \frac{u_*^2}{g} + c_\nu \frac{\nu}{u_*}, \tag{4.194}$$

where c_0 is the Charnock constant, u_* is the friction velocity, g is the acceleration due to gravity, c_υ is a constant, and υ is the molecular viscosity. The first term is the Charnock relation that accounts for high wind (aerodynamically rough conditions), while the second term accounts for low wind (aerodynamically smooth conditions).

4.6 COAMPS Message Passing Interface

The underlying principle in COAMPS programming is for the code to be capable of executing efficiently across vector, parallel, or symmetric multiprocessor (SMP) machines by simply changing run-time options. This requires slightly more overhead through additional arrays and syntax logic, but it pays dividends in flexibility and portability. The COAMPS Message Passing Interface (MPI) code serves this purpose. Originally written in Fortran-77, it now invokes several Fortran-90 (F90) constructs.

4.6.1 Memory management

Memory management for the forecast model is carried out using F90 modules, pointers, and allocatable arrays, as designated in module *coamm_memm.F*. The primary allocatable array *adom* references most of the atmo-

spheric forecast variables. Two-dimensional and three-dimensional arrays are referenced within *adom* using pointers. Similarly, the allocatable array *tracer* is used for passive tracer variables within the *coamm.F* module.

4.6.2 Domain decomposition

The integral design component of the new scalable COAMPS code is the use of horizontal domain decomposition. The decomposition of the entire model domain into subdomains is based on runtime, that is, user-specified values of the number of subdomains in the x- and y-directions (*gridnl* parameters *ndxnam* and *ndynam*). Based on these values, the decomposition technique automatically partitions the data across the nodes of a parallel machine; it closely approximates equal grid points per subdomain for each x- and y-direction. Because of the tight vertical coupling that exists in COAMPS, the decomposition is limited to the horizontal plane.

The user is allowed runtime flexibility in specifying the number of grid points to be used in the halo region using *gridnl* parameter *nbnam*. The halo region is necessary to facilitate finite differencing and data communication with nearest neighbors. Because of fourth-order differencing in COAMPS, two halo points are typically used. This is described in further detail in the subsections that follow.

4.6.2.1 Subroutine *domdec*

Subroutine *domdec.F* is used to set up the domain decomposition, as well as the indices and variables that are computed to expedite communications between processors. The subroutine is accessed in *coamm.F* through the *use domdec* F90 construct.

Several important array indices help define various regions used in the domain decomposition. *maglob* and *naglob* are global indices for the full extent of the domain on a given nest in the x- and y-directions (i.e., (*m, n*)) for the shared memory code. Global indices are the same for all processors.

Local indices vary for each processor, depending on the configuration determined by *ndxnam* and *ndynam*. Indices (*iminf, imaxf*) and (*jminf, jmaxf*) indicate the full extent (including the halo points) for each subdomain in the x- and y-directions. Indices *(imini, imaxi)* and *(jmini, jmaxi)* indicate the computational area for a given subdomain. This is similar to the serial indices *(2, m-1)* and *(2, n-1)*. Note that these indices are simply the full extent indices without the halo points. Thus, *imini(inest) = iminf(inest) + nbnam*, and *imaxi(inest) = imaxf(inest) – nbnam*. Indices *(iminp_nest, imaxp_nest)* and *(jminp_nest, jmaxp_nest)* indicate the physical extent of the subdomains and extend from grid points one to *maglob* and one to *naglob*. Thus, these indices are identical to the computational area indices (*imini*) excluding subdomains that contain the physical boundary of the data. For example, if *imini(inest)* = 2, then *iminp_nest(inest)* = 1. If *imaxi(inest) = maglob(inest)* – 1, then *imaxp_nest(inest) = maglob(inest)*.

To define the two-dimensional extent of data on subdomains, indices *iwidth* and *jwidth* are used, where *iwidth(inest) = imaxf(inest) - iminf(inest) + 1*, and *jwidth(inest) = jmaxf(inest) - jminf(inest) + 1*. The two-dimensional area *ijarea* is the simply the product of *iwidth* and *jwidth*.

Additional indices are used to identify a processor relative to its neighbors. The indices *nnorth_nest*, *nsouth_nest*, *neast_nest*, and *nwest_nest* are set equal to zero. If there is no neighboring processor in the direction of the name of the index, then that index is set to one. For example, *nnorth_nest* = 0 for all processors except those along the top row in the domain decomposition. Thus, *nnorth_nest* = 1 for the northernmost processors (i.e., those having no neighbors to the north). The indices *neigh_north*, *neigh_south*, *neigh_west*, and *neigh_east* are similar to the above indices except they include periodic boundary conditions (if applicable). Thus, *neigh_north* = 1 when it has no northern neighbor. If there is a southern periodic neighbor, then *neigh_north* = 0.

Note that in the forecast model numerics and physics time integration driver, subroutine *amodel.F*, these indices are redefined with shorter names for each nest *nne* as follows:

imin = imini(nne)
imax = imaxi(nne)
jmin = jmini(nne)
jmax = jmaxi(nne)
neast = neast_nest(nne)
nwest = nwest_nest(nne)
nnorth = nnorth_nest(nne)
nsouth = nsouth_nest(nne)
iminp = iminp_nest(nne)
imaxp = imaxp_nest(nne)
jminp = jminp_nest(nne)
jmaxp = jmaxp_nest(nne)

Because of grid staggering, special indices for the *u*- and *v*- wind components are needed as follows:
imaxu = imax - neast
jmaxv = jmax - nnorth

Figure 12 shows a sample of a single nest COAMPS horizontal domain, illustrating the two-dimensional domain decomposition for mass points. The sample domain is one-dimension (i.e., 1:*m*, 1:*n*; where *m=maglob* and *n=naglob*). It has 23×23 grid points in the x- and y-directions, as indicated by the heavy solid box. The domain is decomposed into nine subdomains (P0 to P8) in a 3×3 configuration. The heavy shaded areas indicate the computational region for each subdomain, which extend from indices *(imin:imax, jmin:jmax)*. Thus, no computations are performed on the outermost row or column of data except for the specification of the lateral boundary condition data.

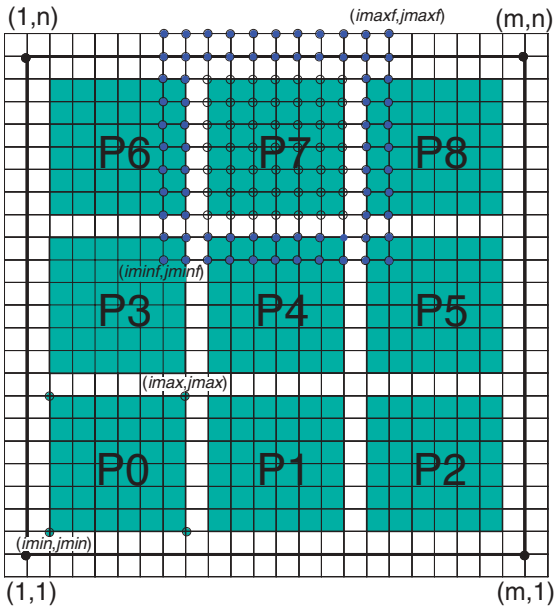


Figure 12.
COAMPS single nest (*m,n*) 23×23 grid point domain for a 3×3 domain decomposition.The computational area of each subdomain is shaded. Open circles indicate computational grid points and solid circles indicate the two halo points for subdomain number 7 (P7).

For this sample domain there are two halo points for each subdomain, illustrated using only subdomain 7 (P7) in Figure 12. The open circles represent grid points within the computational domain of P7. The two rows of solid grid points surrounding the computational points represent the halo region for P7. The extent of P7 with two halo points is from (*iminf:imaxf, jminf:jmaxf*). Note that the halo points extend *nbnam* number of points into the neighboring subdomains.

Loop indices for the MPI code can differ, based on whether the user chooses the runtime option for scalable or serial code (or a combination) based on *gridnl* parameter *loopvecnam*. For serial applications, long vector lengths are desired. However, two-dimensional loop indices are limited by the horizontal extent of each subdomain when using scalable code.

The logic of the new code allows for both types of looping. This is achieved by specifying loop indices that collapse two-dimensional loops into a single dimension. The advantage is vectorization for serial applications, while remaining as two-dimensional indices for scalable applications. The index names, defined in *domdec.F*, are as follows: (1) *(ilof_nest, ihif_nest)* and *(jlof_nest, jhif_nest)* for the full extent; (2) *(ilo1_nest, ihi1_nest)* and *(jlo1_nest, jhi1_nest)* for the interior plus one boundary point; and (3) *(ilo_nest, ihi_nest)* and *(jlo_nest, jhi_nest)* for the computational domain.

For example, when *loopvecnam* = 0 (parallel applications), the full subdomain extent is defined as *ilof_nest(inest) = iminf(inest)*, and *ihif_nest(inest) = imaxf(inest)*. When there is vectorization (i.e., the loops collapse), *loopvecnam* = 1, *ilof_nest(inest)* = 1, and *ihif_nest(inest) = jwidth(inest) * iwidth(inest)*.

Note that subroutine *amodel.F* provides shortened names for each nest *nne* as follows:
ilof = ilof_nest(nne)
ihif = ihif_nest(nne)
jlof = jlof_nest(nne)
jhif = jhif_nest(nne)

ilo1 = ilo1_nest(nne)
ihi1 = ihi1_nest(nne)
jlo1 = jlo1_nest(nne)
jhi1 = jhi1_nest(nne)

ilo = ilo_nest(nne)
ihi = ihi_nest(nne)
jlo = jlo_nest(nne)
jhi = jhi_nest(nne)

4.6.3 Derived datatypes

All MPI communication calls must use a datatype argument. Typically, this is an integer (*MPI_INT*) or a real (*MPI_FLOAT*) type variable. In MPI, however, a powerful tool allows the user to define datatypes (e.g., derived datatypes) for noncontiguous data communication. The derived datatype is composed of a sequence of primitive types (e.g., *INT*, *FLOAT*, or *CHAR*) and a sequence of integer displacements. The displacements define the offset from the initial address for the given data (e.g., zero displacement is the first data element). In COAMPS, the primary uses of derived datatypes are to communicate all vertical levels of halo points between processors and to communicate data from the I/O processor to the appropriate processors. The following example illustrates the use of derived datatypes in COAMPS.

Example 1: To create a derived datatype to communicate all vertical levels of halo points between processors.

This example, from *comflow_ewbord*, illustrates how to communicate halo data in the east-west direction (similar routines exist for the north-south direction). First, one must define the number of blocks (*nlen*) and stride (*nstride*) for the derived datatype as follows:
*nlen = jmaxi(nn) - jmini(nn) + 1 + 2 * numrow*
*nstride = imaxi(nn) - imini(nn) + 1 + 2 * numbord*

numrow is the number of border rows to be communicated, beginning with the innermost. *numbord* is the number of border rows dimensioned in the array. Thus, the number of blocks is the full computational domain in the y-direction plus the halo points to be communicated, and the stride is the computational domain in the x-direction plus the halo points.

The derived datatype is created using *MPI_TYPE_VECTOR* as follows:
call *MPI_TYPE_VECTOR(nlen, numrow, nstride, MPI_REAL8, type_ijborder, 1, ierr_mpi)*

MPI_REAL8 is the old datatype, and the new datatype is called *type_ijborder*. Thus, a new datatype has now been created called *type_ijborder*. It is composed of *nlen* blocks, each block having *numrow* elements of type *MPI_REAL8*, spaced *nstride* elements apart from the start of each block. In two-dimensional space it is clear that the new datatype *type_ijborder* simply defines the east or west halo regions for a given processor for a given level.

Once a datatype has been created, it must be committed before it can be used as follows:
call *MPI_TYPE_COMMIT(type_ijborder, ierr_mpi)*

Now that the horizontal mapping for the new datatype *type_ijborder* is complete, a new datatype that includes all vertical levels can be created. As in the horizontal, the stride (*kstride*) must first be defined as follows:
*kstride = nstride * (jmaxi(nn) - jmini(nn) + 1 + 2 * numbord) * numstridez*

numstridez is the vertical stride (typically set equal to 1). Thus, *kstride* is simply equal to the two-dimensional extent of the data.

The MPI construct *MPI_TYPE_HVECTOR* is used here to create the new datatype *type_border*. *MPI_TYPE_HVECTOR* differs from *MPI_TYPE_VECTOR* only in that the stride between successive blocks is given in bytes, not the number of elements, as follows:
call *MPI_TYPE_HVECTOR(numz, 1, 8*kstride, type_ijborder, type_border, 1, ierr_mpi)*

numz is the number of vertical levels to be communicated. Thus, the new datatype *type_border* is composed of *numz* blocks, each block having one element of type *type_ijborder*, spaced *8*kstride* bytes apart from the start of each block. This datatype represents the desired mapping of all halo points along the east or west boundary and all vertical levels.

Finally the new datatype must be committed as follows:
call *MPI_TYPE_COMMIT(type_border, ierr_mpi)*

4.6.4 Communication between processors

Communication between subdomain processes is achieved using either *MPI-1* or *-2* controlled by *coamnl* namelist parameter *lmpi2 (= true use MPI-2; = false use MPI-1*).

The routine used for COAMPS MPI communications, *comnear.F*, contains several subroutines to control the flow of data between processors. The most frequently used communication routine, *combord.F*, provides for standard border communication of halo points from one processor to neighboring processors. Arguments to *combord* are *(dat, nn, numrow, numz, puv, mtag, lmpi2). dat* is the array whose border (or halo) data is being communicated, *nn* is the nest number, *numrow* is the number of border rows to be communicated beginning with the innermost (each border row is assumed to extend *numrow* elements into the transverse border), *numz* is the number of vertical levels to be communicated, *puv* is the variable type (pressure, u- or v-wind component), and *mtag* is the message tag.

The basic logic for communicating halo regions for both MPI-1 and MPI-2 is to: (1) determine the regions to be sent and received between processors (based upon *numrow*); (2) determine which processors need to send and/or receive data (based upon the neighbor relationships *neigh_north, neigh_south*, etc.); (3) define appropriate derived data types for the halo regions (based on *numrow* and *numz*); then (4) communicate using either MPI-1 or MPI-2.

MPI-1 communications predominately use blocking communication through basic MPI calls to *mpi_send, mpi_recv,* and *mpi_sendrecv*. If MPI-2 communication is used, COAMPS first allocates the necessary symmetric memory in module *mpi2_comm.F*. Then the communication routines use fences to define the communication region (*mpi_win_fence*) and one-sided MPI-2 calls to *mpi_put* and *mpi_get* to communicate halo regions.

The following example, again using subroutine *comflow_ewbord*, illustrates the differences between MPI-1 and MPI-2 when communicating halo data in the east-west direction.

Example 2. To use MPI-1 communications to communicate halo points between processors in the east-west direction.

MPI-1 uses the derived datatypes discussed in Example 1 to communicate between processors. The datatype *type_border* represents the mapping of all halo points along the east or west boundary and all vertical levels. Communication between the appropriate processors is achieved using the neighbor indices (such as *neigh_west, neigh_east*, etc.) to determine who the neighbor processors are, and which processors should send, receive, or send and receive. For example, a processor must communicate with its western neighbor as long as it is not the westernmost processor (assuming nonperiodic conditions). Thus, for *comflow_ewbord*, the easternmost processor will post an *MPI_RECV* from the west:

```
call MPI_RECV(dat(imini(nn)+iaddw_e,jmini(nn)-numrow,1,1),1,
     type_border, nprw(nn), mtag, MPI_COMM_WORLD, mpi_status, ierr_mpi)
```

the westernmost processor will post an *MPI_SEND* to the east:

```
call MPI_SEND(dat(imaxi(nn)+iadde_e,jmini(nn)-numrow,1,1), 1,
     type_border, npre(nn), mtag, MPI_COMM_WORLD, ierr_mpi)
```

and all other processors will post *MPI_SENDRECV* for both the east and west directions:

```
call MPI_SENDRECV(dat(imaxi(nn)+iadde_e,jmini(nn)-numrow,1,1), 1,
     type_border, npre(nn), mtag, dat(imini(nn)+iaddw_e,jmini(nn)-numrow,1,1),
     1, type_border, nprw(nn), mtag, MPI_COMM_WORLD, mpi_status,
     ierr_mpi)
```

MPI-2 communicates first to the east by using the window *iwin1* created in the module *mpi2_comm.F*. The data that will be communicated is copied to array *bufe* for each processor that will communicate with the east and put on symmetric memory *iwin1* using *mpi_put*:

```
call mpi_put
     (bufe,lsr,mpi_real8,npre(nn),itarget_disp,lsr,mpi_real8,iwin1,ierr_mpi)
```

A similar method is used for western communication using window *iwin2*, array *bufw* and *mpi_put*:

```
call mpi_put
     (bufw,lsr,mpi_real8,nprw(nn),itarget_disp,lsr,mpi_real8,iwin2,ierr_mpi)
```

Once the data have been put on symmetric memory using *iwin1* and *iwin2*, the data in arrays *buf1* and *buf2* that are referenced to the windows *iwin1* and *iwin2* in *mpi2_comm.F* are distributed to the correct location on the appropriate processors.

4.6.5 Nesting

4.6.5.1 Fixed nests

An arbitrary number of inner nests can be specified in COAMPS. The only constraint is that the number of grid points must be a multiple of three plus one. This results in a 3:1 reduction in horizontal resolution. There is a similar reduction in time step such that the inner (or child) nest is called three times for each parent time step. Each child nest is decomposed in a similar fashion to that described in Section 4.6.2.

The added complexity arises from the required communications between the parent and child nests. Figure 13 shows the single nest of Figure 12 with a child nest now included. The open circles show the computational area of parent subdomain (P0), and the solid circles depict the two halo points. The heavy solid line through the child nest illustrates the extent of the parent subdomain into the child. Note that because of the 3:1 restriction in nest resolution, parent points coincide with every third child point.

In Figure 14, the child nest is decomposed with the 3×3 configuration. However, COAMPS does allow different configurations for each nest as long as the total number of subdomains is the same. The child nest three (C3) has dimensions of 25×25 grid points, with the two solid grid points surrounding the computational area indicating the halo region. The heavy solid lines through C1, C3, and C4 represent the boundaries of the parent subdomain (P0) shown in Figure 13. For this example, the other parent subdomains are not shown. However, any given child subdomain can overlap with several different parent subdomains.

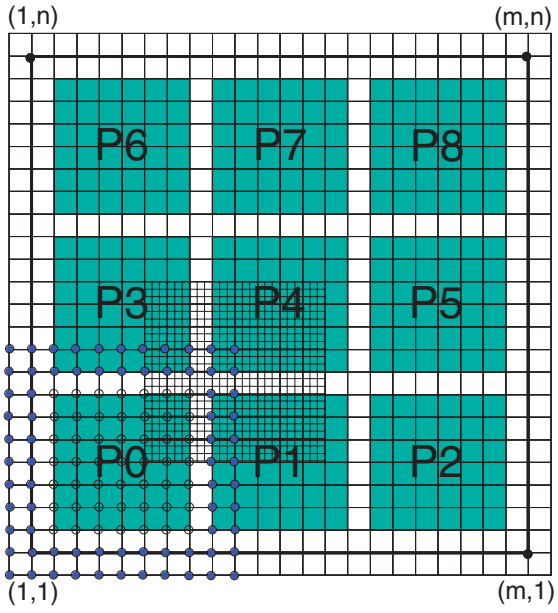


Figure 13.
COAMPS parent nest domain similar to Figure 12, but in addition showing the overlap with a child nest. The open circles show the computational points and the closed circles show the two halo points of P0. The child nest is a 25×25 grid point domain. The heavy solid lines through the center of the child domain represent the outer extent of the parent subdomain for P0 that intersect with the child domain, as shown in Figure 14.

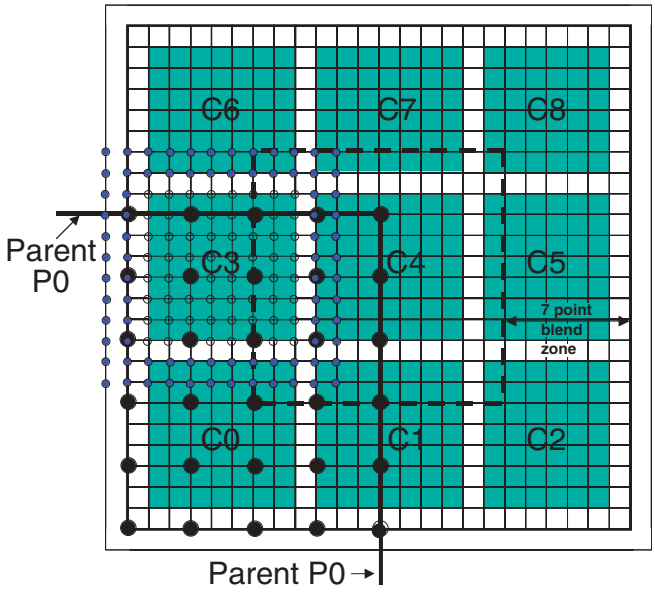


Figure 14.
COAMPS child nest with 25×25 grid points and 3×3 domain decomposition. The heavy solid lines and larger solid circles represent the parent processor (P0) as shown in Figure 13. The smaller open circles are the computational points, and the closed circles are the two halo points of C3. The seven-point blend zone is shown as the area outside the dashed box, on the right-hand side of the figure.

For one-way interactive nesting (two-way interaction is not currently supported in the MPI version), the communication of boundary information is from the parent nests to the child nests at every time step of the parent nest. The data that is communicated and the processors that communicate are determined in modules *bdy_dom.F* and *bdy_comm.F*. Routine *setup_bdy.F* in module *bdy_dom.F* determines the indices for the boundary information from the parent nest communicated to the child. It also allocates space over a user-specified region surrounding the entire child domain, called the blend zone (Figure 14). The blend zone, typically five to seven child grid points wide on the child nest, is determined by namelist parameter *nbdypt*.

The programming practice of using masks is used in the nesting strategy to distinguish regions of the appropriate parent and child nests that need to be communicated. Masks, assigned in routine *maskbdy*, are simply "on or off" switches. Data points for a given subdomain, either parent or child, are assigned mask values of 1 ("on") if they are in the blend zone and 0 ("off") if they are not. Masks for each child subdomain are determined from the number of blend zone points on the perimeter of the subdomain. Parent masks are determined based on the origin reference location and extent of the inner nests relative to the parent nest. After the masks have been determined, each parent subdomain determines whether and where it should send data based on the number of "on" mask values and their relationship to child subdomains. Similarly, child subdomains determine from which parent subdomain they should receive data based on their mask values and their relationship to the parent subdomains.

Once the masks, indices, and communication relationships have been determined, routine *commptoc* (in module *bdy_comm.F*) sets the pointers and allocates space for the data sent and/or received. The data received from a parent processor is stored in a temporary array (or envelope) that resides on each child processor. Each envelope is defined in parent coordinate space and is designed to just cover the domain size of the entire child subdomain. Only those parent points required to compute the horizontal interpolation within the blend zone on the nest boundaries are sent to the envelope array. Offsets are computed to map the arriving parent grid points into the appropriate location within the envelope. Once the parent data points arrive, horizontal interpolation from the coarse mesh data of the parent grid to the fine mesh points of the child grid is then performed using the previously defined masks. As illustrated in Figure 15, the position and size of the envelope are required to be slightly different for interpolating data to the mass and momentum points due to the Arakawa-C grid staggering.

The hatched region in Figure 16 shows the parent envelope for mass points that must be communicated from parent (P0) to child (C3) for the sample horizontal domain. The region contains nine parent grid points that represent the intersection of P0 and the child blend zone on C3. Similar regions exist for each blend zone region on all child subdomains except C4. C4 is an interior subdomain and does not have an exterior blend zone region.

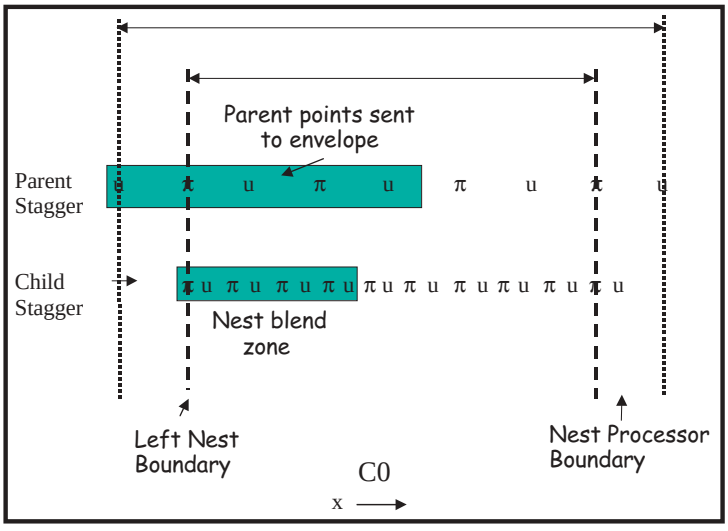


Figure 15.
One-dimensional envelope structure for an Arakawa C-grid.

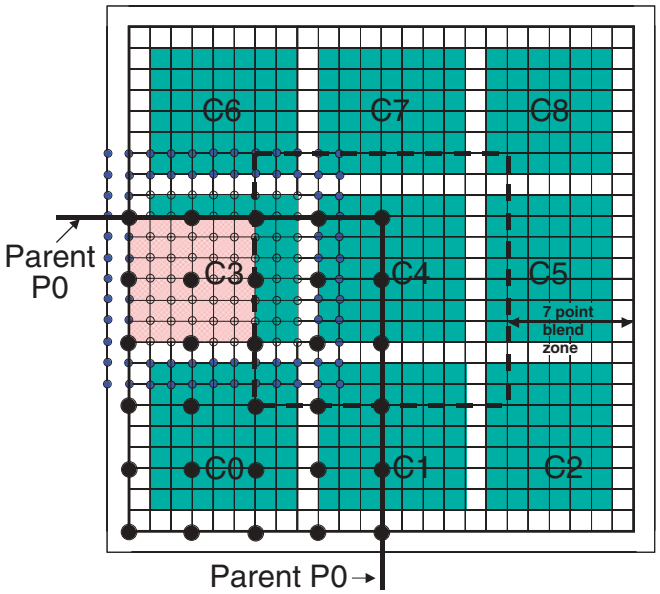


Figure 16.
Sample COAMPS child nest and parent nest similar to Figure 14. The nine larger solid circles in the hatched region are the envelope of parent mass grid points that must be communicated from parent (P0) to child (C3) based on the given seven point child blend zone. In this hatched region, mask values for parent and child nests both equal one.

4.6.5.2 Moving nests

COAMPS has the option to allow the inner nests to move in time (namelist variable *lnmove* = true). By default, *lnmove* for nest one (*lnmove*(1)) must be false (i.e., the outer nest cannot move). Any single innermost nest can move independently by setting *lnmove(inest)* = true. However, if an inner nest is set to move in time (*lnmove(inest)* = true for *inest*>1), then all children nests must move with the parent. They also must remain fixed in their initial position relative to the parent nest. For example, if there are three nests and nest two moves to the northeast, then nest three must also move to the northeast, maintaining its position relative to nest two.

If an inner nest is selected to move, then *coamm.F* will perform initial bookkeeping to ensure that the grids are setup properly. This may be further complicated if data assimilation is used for a nest that has moved during the previous cycle and is in a location other than the designated location at the start of the next cycle. This situation is handled using the move data record *movehd*. Data records at each move time are compared to the data record at the start of the next data assimilation cycle. The data record written closest in time, but not after the current time, is used to determine the grid configuration.

After the grids have been properly set up and initialized, the movement of the selected nest is calculated at each move time according to the value of *imovtyp*. Currently there are two established options for *imovtyp*. If *imovtyp* = 1, then the number of grid points that the selected nest will move in the x- and y-directions (*nmovei*, *nmovej*) must be pre-selected in the namelist by the user for each time given by *itmove* (hours, minutes, seconds). If *imovtyp* = 2, then the selected nest will be moved to a location centered at namelist latitude and longitudes given by (*xlatmov*, *xlonmov*) at time *itmove*.

Several safety checks are related to the movement of the nests. To ensure that the nest does not move too quickly, the nest is not allowed to move more than the number of grid points per time step specified by the namelist variable *movemx*. Also, to avoid MPI communication problems, no nest is allowed to move over a distance farther than the distance of a subdomain in a single time step. This prevents a subdomain from being completely skipped in either the x- or y-direction. To keep the moved nest away from the boundaries, no nest is allowed to move closer than *nbdypt* points to the parent boundary. The following examples illustrate how the namelist can be used to set up moving nest runs for *imovtyp* = 1 or 2.

Example 1:

nnest = 3, *lnmove* = f,f,t,
imovtyp = 1,
itmove = 6000*0, 3, 24, 0,
6, 0, 0,
nmovei = 2000*0, 1, -2,
nmovej = 2000*0, 2, 1,

In this example, only nest three is moved using *imovtyp* = 1. At 3 hours and 24 minutes, nest three will be moved one grid point in the positive x-direction and two points in the positive y-direction. At 6 hours, nest three will be moved two points in the negative x-direction and one point in the positive y-direction.

Example 2:

nnest = 3, *lnmove* = f, t,
imovtyp = 2,
itmove = 3000*0, 5, 12, 0,
8, 52, 0,
xlatmov = 1000*-999.9, 35.4,
36.0,
xlonmov = 1000*-999.9, 235.2,
235.9,

In this example, there are three nests; only nest two has been selected to move. Therefore, both nest two and nest three must move together. COAMPS automatically calculates how many grid points to move nest two from its initial position. For example, at time 5 hours and 12 minutes, nest two will be centered at 35.4 latitude and 235.2 longitude. Then, at time 8 hours and 52 minutes COAMPS will move nests two and three so that nest two will be centered at 36.0 latitude and 235.9 longitude.

The allocation of arrays and pointers for moving nests is similar to that for fixed nests. The moving nest option uses modules *move_dom.F* and *move_comm.F*, similar to the methods found in *bdy_dom.F* and *bdy_comm.F*. At each move time, routine *setup_move* will be called to initialize parent and child arrays and pointers (similar to *setup_bdy*, although it is only called once). All relevant surface, base state, and predictive fields are moved first in the x-direction (if specified) in *move_ew_sfc.F* and *move_ew.F*, and then in the y-direction (if specified) in *move_ns_sfc.F* and *move_ns.F*.

The communication of moved-nest information is similar to that of fixed nest information. At the time a nest is moved, however, the reference points for the parent grid must be reset to allow for the correct information to be communicated to the child nests. In addition, when a child nest is moved, the new region in which the nest has moved must be interpolated from existing information on the parent nest. The interpolated region could span across a variety of parent processors and require additional communications.

4.6.5.3 Delayed nests

The delayed nest option in COAMPS can be used for any child nest (*inest* > 1) simply by specifying the time to start the delayed nest using *idelay* (hours, minutes, seconds). The allocation of arrays, pointers, and communication for delayed nests uses modules *delay_dom.F* and *delay_comm.F* and is similar to that for moved nests. The following example illustrates how to use the delayed nest option.

Example 3:

nnest = 3
idelay = 0, 0, 0,
3, 12, 0,
6, 0, 0,

In this example, there are three nests. Nest two does not start until 3 hours and 12 minutes, and nest three does not start until 6 hours. From 0 hour to 3 hours and 12 minutes, COAMPS runs with one nest. From 3 hours and 12 minutes to 6 hours, COAMPS runs with two nests. After 6 hours, COAMPS runs with three nests.

4.6.6 COAMPS input/output (I/O)

COAMPS processes a tremendous amount of data for any given analysis and forecast cycle. Thus, efficient handling of both input and output data is crucial for optimal computational performance.

4.6.6.1 Data input

Initial and boundary condition data is input to COAMPS using either MPI-1 or MPI-2 constructs; this is user-specified at runtime with the namelist variable *lmpi2*. All two-dimensional input fields are handled in *comin2d.F*. For three-dimensional sigma level data, *cominsig.F* and *cominsigw.F* are used. Boundary condition data is input in *getbc.F*. All of these routines use similar logic.

A designated I/O processor (*npr0*) first reads the entire two- or three-dimensional field into its own local memory. It then distributes the appropriate data to each associated subdomain as specified by the domain decomposition. For MPI-1, this is accomplished through derived data types and blocking *mpi_send* and *mpi_recv*. For MPI-2, the data is first repacked in a contiguous buffer, then *mpi_fence*, *mpi_put*, and *mpi_get* are used to accomplish the task.

4.6.6.2 Data output

The routine used to output COAMPS flat files is called *output.F*. Within *output.F*, four routines are called to output specific types of data: (1) *iosig.F*, which outputs sigma-level data and selected surface fields necessary for data assimilation; (2) *outsfc.F*, which outputs surface fields; (3) *aoutp.F*, which outputs pressure-level fields; and (4) *aoutz.F*, which outputs height-level fields.

Because fields are output on the entire physical grid and computations occur only on the computational grid, the outer boundary points in *outsfc.F*, *aoutp.F*, and *aoutz.F* are handled differently to ensure smooth gradients. This is accomplished using routines *mpibound1.F*, *mpibound2.F*, and *mpibound3.F*. The primary output routine within each of these four output driver routines is called *mpiosig.F*.

The output of COAMPS data within *mpiosig.F* can be handled in two separate ways as specified by the user at runtime (*lmpi2*). If *lmpi2* = true, then COAMPS uses MPI-2 windows to output the data. Each processor uses *mpi_put* to write its portion of the data into the remote memory access area on the I/O processor (*npr0*) as specified by the window. The I/O processor then simply outputs the data. If *lmpi2* = false, then COAMPS uses MPI-2 I/O directives to output the data (note: *mpi_send* and *mpi_recv* are <u>not</u> used with MPI-1). Each processor opens the output file using *mpi_file_open*, then determines the unique location of its data in the output file record using *mpi_file_seek*, and writes its data into that location using *mpi_file_write*. Thus, all processors write to the data file. This contrasts with *lmpi2* = true in which a single I/O processor is used.

5. Aerosol-Tracer Module

5.1 Introduction

A number of naturally occurring and anthropogenic aerosols and chemical species affect U.S. Navy operations, such as weapons systems and navigation, in nearly all parts of the world. Recognizing the need of the Navy to have predictions of mesoscale weather for tropospheric aerosol and chemical conditions, a three-dimensional aerosol-tracer module has been developed within the COAMPS model framework. The module calculates the dynamics and major physics of aerosols and tracers with regard to composition and size-dependent source functions, advection, diffusion, sedimentation, dry deposition, and wet removal of stable and convective precipitation. The module integrates to provide real-time forecasts of mass loading, size distribution, optical depth, and other related property fields. The module framework allows for the incorporation of specific particle-transformation processes and gas-phase chemistry, as needed in COAMPS applications.

5.2 Mass Conservation Equation

The aerosol-tracer continuity equation is derived from the mass flux form of conservation equation, which was introduced by the three-dimensional aerosol model of NASA AMS Research Center (Toon et al. 1988). The general coordinates of the mass flux equation are augmented with the COAMPS terrain-following sigma coordinate. In the sigma formula, z is the vertical altitude, z_{top} is the depth of model domain, and z_{sfc} is the terrain height. Sigma is calculated as

$$\sigma = z_{top}(z - z_{sfc}) / (z_{top} - z_{sfc}). \tag{5.1}$$

The continuity equation becomes

$$\frac{\partial C}{\partial t} + \frac{\partial uC}{\partial x} + \frac{\partial vC}{\partial y} + \frac{\partial (d\sigma / dt + v_f)C}{\partial \sigma} - D_x - D_y - D_\sigma = (C_{src} - C_{snk}). \tag{5.2}$$

C represents mass concentration (kg m^{-3}) as defined by the COAMPS transformed coordinate. The horizontal velocity components of x and y directions (u and v) and the particle gravitational fall velocity v_f are scaled separately by the horizontal map scaling factors at the staggered velocity-grid points and by the vertical σ-coordinate scaling factor ($z_m = (z_{top} - z_{sfc})/z_{top}$). C_{src} is the source and C_{snk} is the sink of aerosols or tracers represented in units of kg m^{-3} t^{-1}. The source concentration term may represent any of the following: injection of aerosol particles, formation of new particles by nucleation, or tracer emission. The sink concentration term may represent a parameterized precipitation removal, surface deposition, or other processes (e.g., heterogeneous nucleation). The terms D_x, D_y, and D_σ all represent subgrid scale turbulent mixing in the x, y, and σ directions. They are defined by

$$D_x = \frac{\partial}{\partial x}\left(K_x \frac{\partial C}{\partial x} \right), \tag{5.3}$$

$$D_y = \frac{\partial}{\partial y}\left(K_y \frac{\partial C}{\partial y} \right), \tag{5.4}$$

$$D_\sigma = \left(\frac{z_{top}}{z_{top} - z_{sfc}} \right)^2 \frac{\partial}{\partial \sigma}\left(K_z \frac{\partial C}{\partial \sigma} \right). \tag{5.5}$$

K_x and K_y are the eddy diffusivities in x and y directions that are scaled by the horizontal map factors at mass grid points. K_z represents vertical diffusion. COAMPS atmospheric model provides u, v, $d\sigma/dt$, K_x, K_y, and K_z fields as well as the thermal dynamic fields that are required for particle terminal velocity, mass production, transformation, and removal processes.

5.3 Aerosol Size Bins

Aerosols must be separated by size categories and particle types to account for aerosol microphysics processes. Size bin distribution, based on a geometrically increasing particle volume, is adapted from the aerosol models of Westphal et al. (1987) and Toon et al. (1988). Radius R_i and volume V_i of the particle centered in the ith volume bin are defined as

$$V_1 = 4\pi R_1^3 / 3, \tag{5.6}$$

$$V_i = (V_{rat})^{i-1} V_1, \tag{5.7}$$

$$R_i = (V_{rat})^{(i-1)/3} R_1, \tag{5.8}$$

where V_l and R_l are the volume and aerosol particle size of the first bin. V_{rat} is the ratio of adjacent bins (i.e., $V_{rat} = V_{i+1} / V_i$). The number of bins decreases when V_{rat} increases. As a result, computation is reduced. Generally speaking, when V_{rat} is greater than or equal to two, the desired bins are generated without reducing the size range covered. Appropriate values for V_{rat} and R_1 depend on the actual size range and bin resolution of interest.

Following the methods of Westphal et al. (1987) and Toon et al. (1988), the bins are centered in volume. The bin upper boundaries (e.g., upper radius edges) are specified as R_{up} using

$$R_{up} = [(2 V_{rat}) / (1 + V_{rat})]^{1/3} R_i . \tag{5.9}$$

The width of bins in radius dR_i and volume dV_i are defined as

$$dR_i = [2 / (V_{rat} + 1)]^{1/3} \, (V_{rat}{}^{1/3} - 1) R_i , \tag{5.10}$$

$$dV_i = [(V_{rat} - 1) / (V_{rat} + 1)] 2 V_i . \tag{5.11}$$

Mass concentration values are stored in a five-dimensional array. The first three dimensions define the model grid space in the x, y, and vertical directions. The aerosol components or tracer species are stored in the fourth dimension, while size bins are stored in the fifth dimension. Passive tracers are special cases of aerosol modeling; they are treated with only one bin and without a size specification.

5.4 Numerical Methods

5.4.1 Time splitting

The three-dimensional tracer continuity equation (5.1) can be integrated forward in time with three equivalent one-dimensional equations. This is referred to as the time-splitting method. The numerical analogs of the one-dimensional equations are simpler than the numerical analogs for the multidimensional equations, making the time-splitting method more advantageous. The multidimensional equations require consideration of cross-space derivatives of various orders (Tremback et al. 1987), which make them more complex than the one-dimensional equations. Another significant advantage of time-splitting is that it can be run as a one-, two-, or three-dimensional model simply by solving different sets of the one-dimensional equations for various modeling tests (Carmichael et al. 1986 and Toon et al. 1988). The source and sink processes may dominate the computational

expense, depending on the complexity of aerosol simulations. Therefore, the continuity equation (5.1) is replaced with the following three one-dimensional equations and one time-tendency equation of C_{src} and C_{snk} as follows:

$$\frac{\partial C}{\partial t} + \frac{\partial uC}{\partial x} - D_x = 0, \tag{5.12}$$

$$\frac{\partial C}{\partial t} + \frac{\partial vC}{\partial y} - D_y = 0, \tag{5.13}$$

$$\frac{\partial C}{\partial t} + \frac{\partial (d\sigma / dt + v_f)C}{\partial \sigma} - D_\sigma = 0, \tag{5.14}$$

$$\frac{\partial C}{\partial t} = (C_{src} - C_{snk}). \tag{5.15}$$

Equations (5.12), (5.13), (5.14), and (5.15) are solved sequentially. The new concentration solved from (5.12) is used to solve (5.13), and so forth.

The sink term in the module can be treated either as implicit or explicit in the calculations. In the discretized form of the implicit method, Equation (5.15) can be expressed as

$$\frac{C_j^{t+1} - C_j^t}{\Delta t} = C_{src} - C_j^{t+1} C_{snk}. \tag{5.16}$$

The *t* and *j* terms are the time and grid indexes, respectively. C_{snk} becomes the lost rate in (5.16) that has a unit of time only (e.g., s^{-1}). Hence, C_{snk} does not need to be multiplied by the scaling factors that are already included in *C* (Toon et al. 1988). The implicit method is preferred over the explicit one because it always results in positive concentrations, provided that C_{src} and C_{snk} have positive values.

5.4.2 Dust aerosol mobilization

Mineral dust, produced by wind erosion over arid or semiarid areas, is a major component of natural atmospheric aerosol. The airborne dust modifies the Earth radiation budget, cloud microphysics processes, and atmospheric chemistry. It also has implications for aircraft operations. For these reasons, the COAMPS aerosol-tracer module incorporates dust mobilization as a type of source function. Dust aerosol mobilization is included with dust microphysics to simulate and predict severe dust storms.

The dust emission formula used in the module is derived from field data by Westphal et al. (1987 and 1988), Gillette and Passi (1988), and Nickling and Gillies (1993). The formula describes the vertical dust flux *F* in proportion to the friction velocity raised to the fourth power (or the square of surface wind stress) as

$$F = f A u_*^4 \quad \text{when} \quad u_* \geq u_{*_t}. \tag{5.17}$$

The vertical dust flux has units of kg m^{-2} s^{-1} particles. The term *f* is the fraction of the model grid box that is dust erodible, *A* is a dimensional constant, and u_{*_t} is the threshold friction velocity. Liu and Westphal (2001) conducted numerical studies by comparing the u_*-driven flux (5.17) with dust observations and also with another dust emission formula derived from the wind tunnel experiment of Gillette (1978). The latter assumes the vertical dust flux is proportional to the third power of surface wind speed. Equation (5.17) accounts for both wind shear and thermal stability effects on wind stress. It was proven to be more realistic and preferable than the wind-driven flux formula.

Considering the dust flux for particles with radii less than or equal to 50 mm, A is set to 1.42×10^{-5} (Nickling and Gillies 1993). Based on the various values of U_{*t} for different land categories in Gillette and Passi (1988), 0.60 ms^{-1} was chosen for the default U_{*t} value in the model. However, it may be land-type dependent in the model as required for specific case studies. According to Marticorena and Bergametti's particle model (1995), this value is the threshold for 90% of the particles in a measured mass size distribution from a sandblasting experiment (Alfaro et al. 1997).

The fractional coverage of dust-erodible lands within each grid box is derived from a 1-km-resolution global land cover characteristics database, produced by the U.S. Geological Survey's Earth Observation System Data Center, the University of Nebraska-Lincoln, and the Joint Research Center of the European Commission. Eight land-use types are identified as equally dust erodible. The erodible fraction is determined by calculating the percentage of 1×1-km land-use elements that are considered erodible. A land survey by Clements et al. (1957) shows that only 13%, on average, of the erodible lands within each grid box are capable of emitting dust. Therefore, the erodible fraction is multiplied by a factor 0.13 to yield the f in Equation (5.17). Dust emission is further restricted to the area where the predicted soil moisture is less than or equal to 0.3 (fraction by volume). The choice of dust emission value is empirical. It is based on a comparison of the distribution of springtime dust storm reports and the climatological ground wetness data used by COAMPS, and averaged over the Asia dust source areas in the springtime (Liu and Westphal 2001).

5.4.3 Particle size distribution

Aerosol particle size distribution by mass is calculated immediately following the aerosol mass production or mass flux process. Because optical effects of aerosols depend highly on their size distributions, accurate modeling of particle sizes is critical for modeling predictions. In this module, two generalized types of mass distribution formulae are used for different cases. The first one is the lognormal mass distribution with multiple modes expressed by Marticorena and Bergametti (1995) as

$$\frac{dM}{d\ln R} = \sum_{j=1}^{n} \frac{M_j}{\sqrt{2\pi}\,\ln(\sigma_j)} \exp\left(\frac{(\ln R - \ln R_{mj})^2}{-2\ln^2(\sigma_j)}\right). \tag{5.18}$$

The M term is the mass generated from sources, R is the particle radius, j refers to the jth mode, n is the total number of modes, M_j is the mass fraction of particles for mode j, R_{mj} is the mass median radius, and σ_j is the geometric standard deviation of the jth mode. This lognormal function, which is particularly convenient to model single, two, three, or more modal distributions, requires a reasonable number of parameters. The values of parameters M_j, R_{mj}, and σ_j for each option can be directly specified based on field and laboratory experiments. The model default is bimodal lognormal distribution with the parameters predicted by the sandblasting model of Alfraro et al. (1997).

The second type of mass distribution is the power-law form adapted from Westphal et al. (1987) as follows:

$$\frac{dM}{d\log R} = cR^{\nu}. \tag{5.19}$$

The c term is a constant coefficient and ν is a free parameter that determines the slope of the straight power-law distribution. c is obtained by integrating mass over the entire size range, given the total mass produced, the smallest particle radius, and the largest particle radius. The n term is assigned a value of 1.2, following the method of Westphal et al. (1987). This simple distribution is advantageous; any deviation in the predicted distributions from the initially straight power-law distribution can easily be detected.

5.4.4 General emission function

When the focus of a numerical study is transport or dispersion of passive tracers and other scalars, a generalized source function representing C_{src} in Equation (5.15) is used. This allows the model to make continuous and instantaneous emissions from a single grid point or from a group of grids to generate plumes, puffs, or clouds over time and in space. The user's specifications in the namelist variables determine the type of emission that will be used in the model.

Letting an emission occur at a single grid point creates a sharp gradient in the advected scalar field near the source; this is the closest approach to a real-world situation. Such sharp gradients can be resolved by using an appropriate high-order accurate transport scheme in an Eulerian model, as demonstrated by Liu and Carroll (1996). By choosing Bott's mass conservative flux-form advection scheme, discussed below, this model is able to perform single grid-point emissions for plumes and puffs.

When an emission comes from a group of grid points (i.e., emission in a two- or three-dimensional space) a cloud shape of two- or three-dimensional volume is constructed as round or oval. The cloud size depends on the horizontal and vertical radii. Emission strength of C_{src} at any grid point (i.e., i, j, and k) within the cloud is calculated by using a cosine function scaling. This scaling produces C_{src} values where the highest point is in the cloud center and zeros at the cloud edges. The spatial distribution of mass concentration within the cloud is symmetrical to the cloud center and the total emitted mass is conserved.

5.4.5 Advection and turbulent mixing

The treatment of advection is crucial for numerically solving all types of continuity equations. For the transport of aerosols and tracers, the requirements of an advection scheme are particularly high because of low concentrations and positive definitions. Therefore, a high-order accurate scheme with the following properties is required: positive definite, mass conservative, dispersion free, diffusion minimized, and computationally efficient. Bott's forward-in-time, positive-definite, area-preserving flux-form advection algorithm (Bott 1989a, 1989b, and 1992) satisfies the requirements for the advection component of the aerosol-tracer module. It also accounts for COAMPS' staggered grid structure.

To make the fitting curves approach real field distributions, Bott's algorithm performs a polynomial fitting to the advected variable in each grid box. The fluxes at grid-box boundaries are integrated upstream from the fitting curves. The coefficients of the polynomial interpolation for each grid box are obtained by assuming the area under the fitting curve is preserved. A weighting flux treatment is conducted to minimize both amplitude and phase errors, and achieve mass conservation and positive definition. The first-degree polynomial fitting gives rise to a (l+1)–order accurate advection scheme. Considering the balance between numerical accuracy and computational cost, Bott's fifth-order accurate scheme with the fourth-degree polynomial fitting is preferred. Therefore, the model was designed to account for advections in the horizontal component, constant grid spacing, and expanding grid spacing in the vertical component. To upgrade the advected variable at one grid point, six or seven grid-points are needed to solve the advection in this algorithm. Bott's seventh-order accurate scheme, using sixth-degree polynomial interpolation, is also programmed into the model as an alternate choice for aerosol or tracer advection.

Turbulent mixing in the vertical (Equation (5.5)) component is solved with the Turbulent Kinetic Energy (TKE) closure (see discussion in Section 4.5), using the eddy exchange coefficient (K_z) generated from the TKE equation in COAMPS. The horizontal eddy diffusivities K_x and K_y (Equations (5.3) and (5.4)) are solved with either the Smagorinsky scheme or large eddy simulation. Both the explicit and implicit numerical diffusion methods are available to solve Equations (5.3), (5.4), and (5.5). The implicit method, involving a tri-diagonal matrix in the global scale, is numerically stable and preferred in the shared memory version of the tracer code. Conversely, the explicit method does not require any global-scale interpolation. Therefore, the explicit method is a reasonable choice for the distributed memory code using the Massage Passing Interface (MPI, Section 4.6).

5.4.6 Sedimentation, dry deposition, and wet removal

Aerosols are removed from the atmosphere by gravitational settling (sedimentation), turbulent mixing at the surface (dry deposition), and washout by precipitation (wet deposition). These physical processes are important for all types of aerosols.

Particle's terminal settling velocity v_f, in units of m s^{-1}, is size-dependent. It is calculated according to the Stokes Law a

$$v_f = \frac{2 R_p \rho_p g C_c}{9 \nu \rho_a} 10^{-5}. \tag{5.20}$$

R_p is the particle radius (μm), ρ_p is particle density (g cm^{-3}), g is gravity (cm s^{-2}), C_c is the Cunningham correction factor, ν is kinetic viscosity of air (cm^2 s^{-1}), and ρ_a is air density (kg m^{-3}). C_c is defined in terms of the mean free path of air molecules (λ in units of mm) as

$$C_c = 1.0 + \frac{\lambda}{R_p}\left(1.257 + 0.4 \exp\left(-1.1 \frac{R_p}{\lambda}\right)\right). \tag{5.21}$$

The values of λ and ν care found in Seinfeld (1986).

Dry deposition for dust is calculated based on the downward mass flux to the surface at the deposition velocity v_d. Mass flux caused by deposition F_{dep} is defined by

$$F_{dep} = -v_d C_{surface}. \tag{5.22}$$

The deposition is modeled as particles are removed from the bottom model layer. In general, the deposition velocity v_d has considerable variability due to variations in meteorology and surface characteristics. The v_d calculation for dust particles is derived with surface wind stress and surface wind speed (Stull 1988) as

$$v_d = u_*^2 / U_{10}. \tag{5.23}$$

U_{10} is the wind speed at a 10-meter height. In practice, dry deposition and sedimentation are calculated together by adding v_d to the terminal velocity v_f for the bottom model layer.

A separate scheme of v_d is used in the model for other aerosol species. The scheme depends on two types of surfaces: open water and land. The dry deposition on open water, provided by the Slinn and Slinn (1980) formula, assumes a particle size range with a mass mean radius near 1 μm as

$$v_d = 1.3 \times 10^{-3} U_{10}. \tag{5.24}$$

For land surfaces, dry deposition is calculated using the methods of Walcek et al. (1986) and Christensen (1997) using

$$v_d = \begin{cases} u_* / a\left(1 + (-300 / L)^{2/3}\right) & \text{for } L < 0 \\ u_* / a & \text{for } L > 0 \end{cases}. \tag{5.25}$$

The a term is a free parameter (generally equal to 500) and L is the Monin-Obukhov length.

Aerosols can be scavenged out of the atmosphere by absorption into cloud droplets within clouds (washout) or by collision with raindrops below clouds (rainout). The scavenging rate of washout is calculated using convective precipitation, while that of rainout is related to stable precipitation. A parameterization of scavenging rate (L in units of s^{-1}) for both washout and rainout is obtained from Pruppacher and Klett (1978). Their formula was derived by assuming an upper limit on particle scavenging within and below clouds and the wet removal being independent of particle sizes and droplet sizes as

$$\Lambda = 3.0\, R^{0.16} / H. \tag{5.26}$$

R is the rain rate of convective or stable precipitation (mm hr^{-1}). When using Equation (5.26) for washout scavenging, H represents the cloud depth (m). Since convective rainfall in COAMPS is a parameterized constant in the vertical component, a constant washout rate Λ is obtained and applied to all the model grid layers between the top and the bottom of a cloud. When using (5.26) for rainout scavenging, H is the vertical grid layer size (m). The rainout rate Λ is then calculated for each grid layer below the cloud using the stable rainfall defined at the grid height.

5.4.7 Aerosol optical thickness

To estimate the direct effects of aerosol size distribution on the scattering and absorbing of solar radiation, the optical thickness of aerosol particles must be determined. Optical thickness is one of the most important optical properties used in studies of aerosol interactions with the global and regional environment. Optical thickness τ_i of a size bin i is calculated in the discretized form to the first order with extinction only as

$$\tau_i = \sum_{k=1}^{n} (Q \pi R_i^2 N) \Delta z_k , \tag{5.27}$$

$$M_p = \frac{4}{3} \pi \rho_p R_i^3 . \tag{5.28}$$

N (m^{-3}) is the number of particles in a unit volume. R_i the representative particle radius of the ith bin. Q is the extinction geometry factor that varies with wavelength, aerosol species, and relative humidity. Δz_k is the layer thickness, and n is the number of model vertical levels. Particle mass M_p is calculated with particle density ρ_p and radius R_i. Substituting the πR_i^2 term in Equation (5.27) with Equation (5.28) and converting the unit of ρ_p from kg m^{-3} to g cm^{-3}, and R_i from m to μm yields Equation (5.29) for a particular wavelength:

$$\tau_i = \sum_{k=1}^{n} \left(\frac{750 Q C}{R_i \rho_p} \right) \Delta z_k , \tag{5.29}$$

where C ($= M_p N$) is the aerosol concentration (kg m^{-3}). The total optical thickness τ is obtained by adding τ_i of all size bins. The extinction coefficient Q is normally taken at a wavelength around 0.53 μm (visible light) for most aerosols. This is because the influence of aerosols on radiation is most significant at visible light.

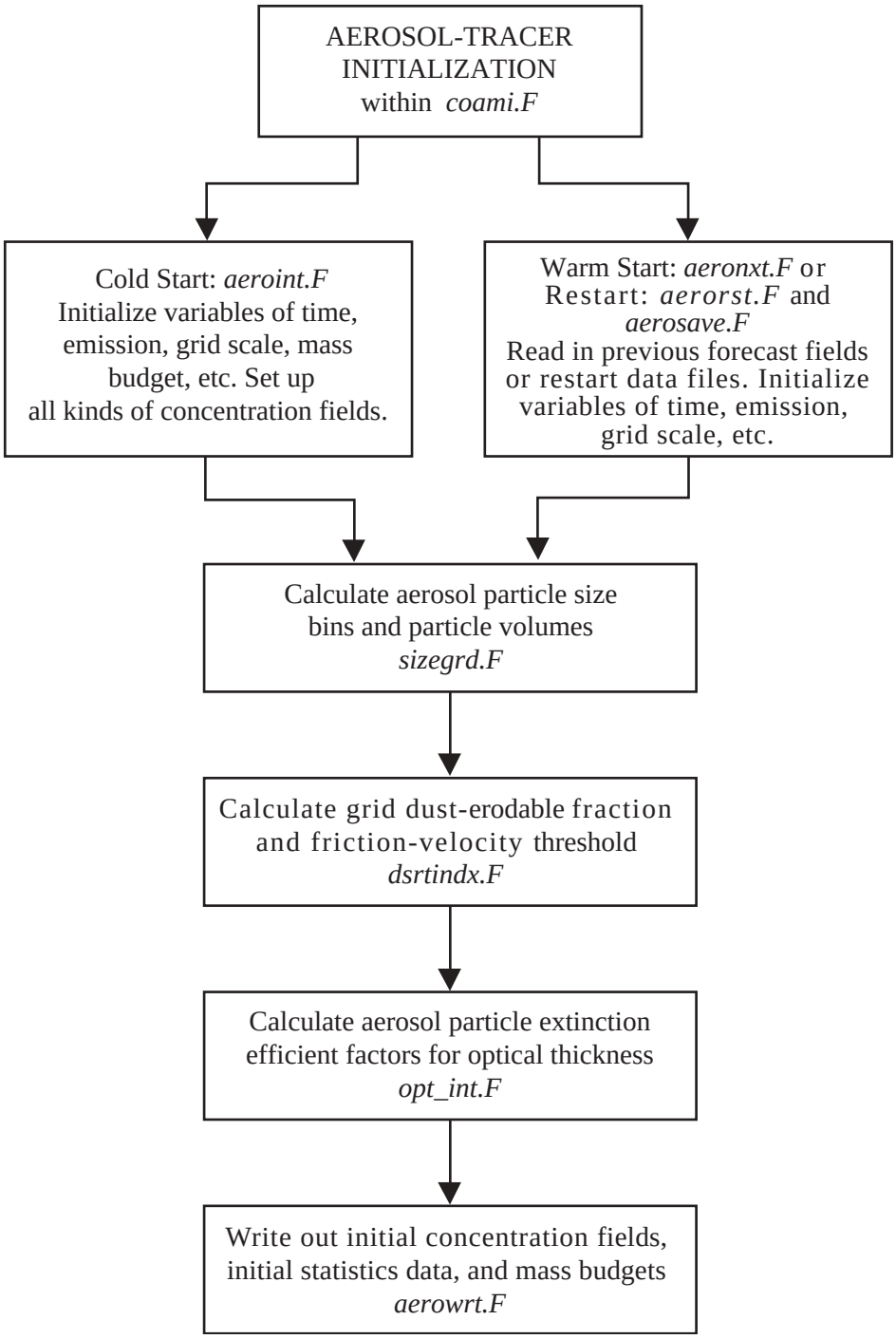
5.4.8 Grid nest interactions

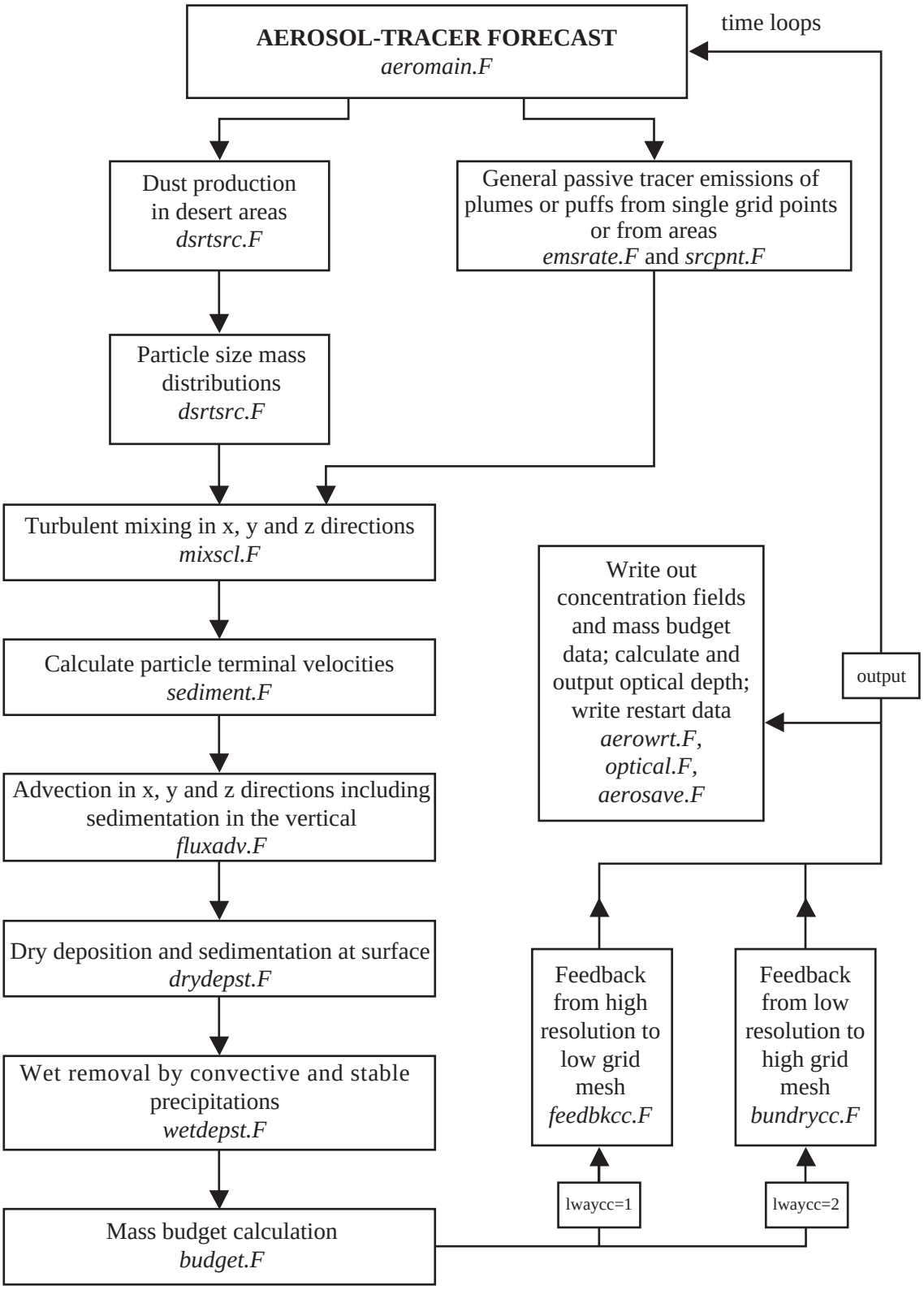
The aerosol-tracer module was developed to use the same nesting procedures as the COAMPS atmospheric model. One important capability of COAMPS is multiple grid nesting. For the aerosol and tracer transport, one- or two-way interactions between high- and low-resolution grid are available. The interaction methods are determined by the namelist parameter *lwaycc*, specified by the user. For the two-way aerosol-tracer interaction, feedback from high- to low-resolution grids and feedback from low- to high-resolution grids are considered.

The feedback from high- to low-resolution grids is examined first. The concentration field in the high-resolution domain is passed to the low-resolution grid points in the overlapping area. This is handled through bilinear averaging of nine high-resolution grid values to one, corresponding low-resolution grid value after every time-step integration. The nesting approach is very useful in simulating long-range transport of aerosols or tracers. The smaller high-resolution grid can be used to accurately predict the mass productions from high-resolution source functions. The low-resolution grid with a larger domain supports the long-range transport of aerosols downwind but with lesser computational cost. Simulations with the high-resolution source production show that continuous updates of the upstream aerosols produce more accurate, low-resolution downstream transport predictions than the high-resolution grid.

The feedback from the low-resolution to the high-resolution grid is also important. Concentrations at the low-solution grid points that surround the lateral boundaries of the high-resolution domain are taken as the boundary in-flow mass fluxes to the high-resolution grid points. The low-resolution data are factored into calculations of the horizontal advection for the high-resolution domain on the lateral boundaries. This feedback is useful when investigating a particular high-resolution area with detailed aerosol microphysics and size distributions. Any upcoming or returning mass flows across the boundaries of the high-resolution domain would contribute to the aerosol's evolution in the inner region similar to the process that occurred in nature. The size of the high-resolution domain is limited by computational cost.

5.5 Program Flow Chart




AEROSOL-TRACER FORECAST
aeromain.F
time loops
Dust production in desert areas
dsrtsrc.F
General passive tracer emissions of plumes or puffs from single grid points or from areas
emsrate.F and srcpnt.F
Particle size mass distributions
dsrtsrc.F
Turbulent mixing in x, y and z directions
mixscl.F
Write out concentration fields and mass budget data; calculate and output optical depth; write restart data
aerowrt.F, optical.F, aerosave.F
output
Calculate particle terminal velocities
sediment.F
Advection in x, y and z directions including sedimentation in the vertical
fluxadv.F
Dry deposition and sedimentation at surface
drydepst.F
Feedback from high resolution to low grid mesh
feedbkcc.F
Feedback from low resolution to high grid mesh
bundrycc.F
Wet removal by convective and stable precipitations
wetdepst.F
lwaycc=1
lwaycc=2
Mass budget calculation
budget.F

5.6 Namelist Variable Definitions

a. Variables in COAMPS namelist "atmosnl.h"

Name	Type	Description	Default Value
iaero	Integer	Turn on/off aerosol-tracer module. = 1, run aerosol-tracer = 0, do not run.	0
mspc	integer	Number of tracer species (used when trceflg = t in aeronl.h)	1
mdust	integer	Number of size bins in dust component (used when dustflg = t in aeronl.h)	1
lgrdall	logical	Flag to control grid nests. = true, run aerosol-tracer on all COAMPS grid nests (variable nestcc is then ignored) = false, run aerosol-tracer on one grid nest only specified by nestcc	True
nestcc	integer	Specify which grid nest to be run for aerosol-tracer (used with lgrdall = false). nestcc = 1, or 2,or 3, etc.	2

b. Variables in aerosol-tracer namelist "aeronl.h"

Name	Type	Description	Default Value
aerotme0 (3)	integer array	Beginning time (hr, min, sec) of the aerosol-tracer integration. ($0 \leq$ aerotme0 $\leq$ COAMPS forecast time)	0,0,0
kcctyp	integer	Type of output time. = 1, output at specified time = 2, output at a frequency.	2
kccfrq (3)	integer array	Frequency or time interval of output (hr, min, sec).	3,0,0,
kcctime (3,kcctot)	integer array	Specify output time (hr, min, sec) with *kcctot* number of output.	*kcctot* = 500
ccycle	logical	Type of model start, cold or warm. = false, cold start = true, warm start	True
lwaycc	integer	Type of grid nest interaction. = 0, no grid nest interaction = 1, feedback from high-resolution grid nest to low-resolution grid = 2, feedback from low-resolution grid nest to high-resolution grid. =3 two-way, including 1 and 2.	0
nadvtyp	integer	Choice of advection schemes used for transport. = 1, Bott's 3rd-order flux-form advection = 2, Bott's 5th-order flux-form advection = 3, Bott's 7th-order flux-form advection	1
trceflg	logical	Flag of passive tracer simulation/forecast (each tracer/species has only one emission source) = f, turn off tracers = t, turn on tracers	T

b. (continued). Variables in aerosol-tracer namelist "aeronl.h"

Name	Type	Description	Default Value
nintcc	integer	Type of tracer concentration initialization. = 0, zero initial concentration field = 1, user provides initial concentration field (used when trceflg = t)	0
irrgsrc	integer	Type of tracer emission source setup, used when (trceflg = t) = 0, regular setup by given cntlat, cntlon, cnthgt, etc. = 1, user provides specific or real case emission-related data	0
cntlat (mspc)	real array	Latitudes of tracer emission site centers, used with trceflg = 1 and irrgsrc = 0.	30.
cntlon (msrc)	real array	Longitudes of emission site centers, used with trceflg = 1 and irrgsrc = 0.	60.
cnthgt (msrc)	real array	Heights (m) of emission site centers above the ground, used with trceflg = 1 and irrgsrc = 0.	10.
ntems (mspc)	integer array	Number of emission-strength changes in time for instantaneous and continuous releases, used with trceflg = 1 and irrgsrc = 0 or 1.	1, continuous emission with constant strength
emstime (3,ntmesX mspc)	integer array	Time list (hr, min, sec) at which emission strength changes for ntems numbers, used with trceflg = 1 and irrgsrc = 0 or 1.	0,0,0
emstrng (ntemsX mspc)	real array	Emission strength or flux (kg/sec) at each time for each species. Used with emssflg = 1 and irrgsrc = 0 or 1.	5.0
ngrdems (mspc)	integer	Geometry type of emission site of each tracer, used with trceflg = 1 and irrgsrc = 0. = 1, emission from single grid volume = 2, emission from a multiple grid volume, i.e. a 3-d round- or oval-shape cloud	1
hhrad (mspc)	real array	Horizontal radius (m) of cloud-shape volume for tracers with ngrdems = 2.	5000.0
zzrad (mspc)	real array	Vertical radius (m) of cloud-shape volume for tracers with ngrdems = 2.	300.0
mbgtrce	integer	Detailed passive tracer mass budget calculation when trceflg = t. = 0, no tracer budget calculation = 1, calculate tracer budget	0
dustflg	logical	Flag of mineral dust simulation/forecast	False
mbgdust	integer	Detailed dust mass budget calculation when dustflg = t = 0, no dust budget calculation = 1, calculate dust budget	0
lmsload	logical	Flag of mass load (vertical integral) calculation and output (mg/m**2), used with all aerosol/tracer components = true, yes = false, no	false

b. (continued). Variables in aerosol-tracer namelist "aeronl.h"

Name	Type	Description	Default Value
ldosage	logical	Flag of mass dosage calculation and output (mg*min/m**3), used with all aerosol/tracer components. = true, yes = false, no	false
ldrydep	Logical	Flag of aerosol/tracer dry deposition calculation, used with all aerosol/tracer components. = true, yes = false, no	false
lwetdep	logical	Flag of aerosol wet deposition calculation, used with all aerosol components. = true, yes = false, no	false
lsedimn	logical	Flag of aerosol sedimentation calculation, used with all aerosol components. = true, yes = false, no	false
lemsflx	logical	Flag of aerosol emission/production flux output (mg/m^2/s), used with all aerosol components.	false
loptic	logical	Flag of aerosol optical thickness calculation and output, used with all aerosol components. = true, yes = false, no	false

5.7 Example Run Script

This section provides an example of setting up aero-tracer namelist variables (bold font). In the following example, the model simulates plumes and puffs by making continuous and instantaneous releases at three emission sites and changing the emission strength at following specified forecast time.

&gridnl

kka = 30,
ma = 71,103,
na = 51,79,
nnest = 2,
iref = 36,
jref = 26,
ii = 1,18,
jj = 1,13,
rlat = 40.0,
rlon = -97.0
alnnt = -97.0
phnt1 = 60.0,
phnt2 = 30.0,
delx = 81000.0,27000.0,
dely = 81000.0,27000.0,

&end

&atmosnl

_____

iaero = 1, --- turn on aerosol/tracer module

mspc = 3, --- three tracers (used when trceflg=t in aeronl.h)

mdust = 1, --- one size bin in dust (used when dustflg=t in aeronl.h)

lgrdall = t, --- run aerosol/tracer on all model grids (when lgrdall=t, nestcc is ignored)

nestcc = 2, --- run aerosol/tracer on grid 2 (used only when lgrdall=f)

&end

&aeronl

aerotme0 = 0,0,0, --- beginning time of aerosol/tracer module (normally set to the earliest tracer emission time when trceflg=1).

kcctyp = 2, --- output aerosol/tracer fields at a specified frequency.

kccfrq = 3,0,0, --- output every 3 hrs.

ccycle = t, --- warm start. If previous forecast fields are not found, ccycle is automatically set to false for cold start.

lwaycc = 0, --- no grid nest interaction.

nadvtyp = 1, --- use 3rd-order Bott's flux-form advection scheme.

trceflg = t, --- turn on passive tracer simulation/forecast.

cntlat = 42., 40., 38., --- latitudes of 3 tracer emission sites.

cntlon = -105., -97., -90., --- longitudes of 3 tracer emission sites.

cnthgt = 10., 10., 10.,

*ntems** = 1, 2, 3, --- numbers of emission-strength change in time for each tracer.

*emstime** = 1, 32, 0, --- emission beginning time of 1st tracer (1st tracer generates a plume)

3, 40, 0, --- 1st emission time of 2nd tracer

3, 44, 0, --- 2nd emission time one time-step later (=4 min) (2nd tracer generates a 4-min puff)

5, 20, 0, --- 1st emission time of 3rd tracer

5, 32, 0, --- 2nd emission time 3-time-step later of 3rd tracer

13, 12, 0, --- 3rd emission time of 3rd tracer (3rd tracer generates a 12-min puff first then generates a plume)

*emstrng** = 100.0, --- constant emission strength of 1st tracer (kg/s/site)

800.0, --- 1st emission strength of 2nd tracer

0.0, --- 2nd zero emission one time step later of 2nd tracer

200.0, --- 1st emission strength of 3rd tracer
0.0, --- 2nd zero emission 3-time-step later of 3rd tracer
120.0, --- 3rd emission strength of 3rd tracer

ngrdems = 1, 1, 1, --- emission from single grid volume for all tracers.
ldosage = f, --- no mass dosage calculation and output.
lmsload = t, --- have mass load output.
ldrydep = t, --- have try deposition for tracers.
&end

[*Meanings of the above *ntems*, *emstime* and *emstrng* specification:

For the 1st tracer, specifying one emission 100 kg/s at 1:32:00 means the starts at this time and continues emitting at constant strength throughout the forecast, generating a plume.

For the 2nd tracer, specifying two emissions of 800 kg/s at 3:40:00 and 0.0 kg/s at 3:44:00 means that emission starts at 3:40:00 and stops 4 minutes later. This 4-min emitting generates a puff.

For the 3rd tracer, specifying three emissions of 200 kg/s at 5:20:00, 0.0 kg/s at 5:32:00, and 100 kg/s at 13:12:00 means that emission starts at 5:20:00 and stops 12 min later, which generates a 12-min puff, then it starts again at 13:12:00 and continues emitting to the end of forecast, which generates a plume.

(For an instantaneous release, set the time increment between the 1st and 2nd emission to the time step of grid 1 if lgrdall=t or to the time step of the nest (nestcc) if lgrdall=f.)]

5.7.2. Aerosol-tracer output fields

3-d tracer_concentration (mg/m^3) output in species in sigma levels:
cccc or *concen*

3-d total_tracer_concentration (mg/m^3) summed all species in sigma levels:
tccc or *totcen*

3-d tracer_boundary_concen (kg/m^3) output in species in sigma levels:
bdcc or *bndcon*

3-d tracer_mass_dosage (km*min/m^3) output in species in sigma levels:
csum or *consum*

2-d tracer_mass_load (mg/m^2) output in species at surface level:
ccld or *ccload*

3-d dust_mass_concentration (mg/m^3) output in bins in sigma levels:
dscc or *dustcc*

3-d total_dust_concentration (mg/m^3) summed all bins in sigma levels:
tdsc or *tdstcc*

3-d dust_boundary_concen (kg/m^3) output in bins in sigma levels:
dsbc or *dustbc*

3-d dust_mass_dosage (kg*min/m^3) output in bins in sigma levels:
dsum or *dstsum*

2-d total_dust_mass_load (mg/m^2) summed all bins output at surface level:
dsld or *dsload*

2-d total_dust_production_flux (kg/m^2/s) at surface level:
dsfx or *dstflx*

3-d dust_extinction_coeff (1/m) output in bins in sigma levels:
dsex or *dstext*

3-d total_dust_extinction_coeff (1/m) summed all bins in sigma levels:
tdex or *tdsext*

2-d dust_optical_depth output in bins at surface level:
dsop or *dstopt*

2-d total_dust_optical_depth summed all bins output at surface level:
tdop or *tdsopt*

2-d dust_erodible_fraction (0.0-1.0) output at surface level:
dsfc or *dstfrc*

2-d dust_ustar_threshold output at surface level:
fvel or *fvlmax*

[Default output from aerosol/tracer module is always 3-d mass concentration fields in species or bins and the total mass concentration summed over all species or bins in sigma levels. The flat file names look like:

3-d mass concentration (mg/m^3)
concen_sig_031050_000001_2a0103×0079_2001040712_00230000_fcstfld
concen_sig_031050_000002_2a0103×0079_2001040712_00230000_fcstfld
concen_sig_031050_000003_2a0103×0079_2001040712_00230000_fcstfld
2-d mass load (mg/m^2)
ccload_sfc_000000_000001_2a0103×0079_2001040712_00230000_fcstfld
ccload_sfc_000000_000002_2a0103×0079_2001040712_00230000_fcstfld
ccload_sfc_000000_000003_2a0103×0079_2001040712_00230000_fcstfld
|
|----> *these 1, 2, 3 indicate tracer species 1, 2, 3*]

5.8 Log File Checkout

When running model simulations or forecasts or when conducting model tests, it is important to have quick and easy examination of the results. This module provides two efficient ways for the user to check out model results for mass concentration fields.

5.8.1 COAMPS forecast log file

The subroutine *aerowrt.F* conducts all of the aerosols and tracers output. At each output time, *aerowrt.F* calculates the: (1) total mass (kg) injected into the model domain since the beginning of run; (2) total mass integrated through the model domain at the output time; and (3) mean, maximum, and minimum values of mass concentration field for each species in each grid nest. If the optical thickness variable is set to true, *aerowrt.F* also calculates the mean, maximum, and minimum values of optical thickness field for each species, including the total over all of them.

By default, the results are output to the COAMPS forecast log file. The data can provide model-users the most primary information about the program execution. For example, users would have instant access to the amount of mass generated and the amount of mass lost through boundaries. Using these statistics, the user can determine the approximate concentration field distribution.

If a problem should occur in the aerosol-tracer module, the data in the log files may look erratic. This might indicate errors in either the mass production or mass concentration integration. In this case, the user may troubleshoot by using the detailed mass budget calculation described below.

5.8.2 Mass budget calculation

The mass budget calculation is a very important part of aerosol-tracer modeling because mass conservation is highly demanded. Mass changes caused by numerical approximations in dynamic and physical processes should be within acceptable ranges. Model performance must be monitored through the mass budget calculation following individual process integration at each time step in each grid nest. Monitoring includes the following items:

1) total mass emitted or generated at the current time step,
2) total mass within the model domain,
3) boundary outflow mass in the horizontal and vertical advection,
4) boundary inflow mass in the horizontal and vertical advection,
5) mass changes resulting from advection scheme in x, y, and z directions,
6) mass changes resulting from turbulent mixing in x, y, and z directions,
7) mass lost to the surface by dry deposition and sedimentation,
8) mass removal out of the air by wet deposition, and
9) net mass change after all above modeling processes.

Item 9 is obtained by adding items 2 through 8, subtracting from item 1, then dividing by item 1 (i.e., (9) = { (1) - [(2)+(3)+(4)+(5)+(6)+(7)+(8)] }/(1)). This value can be considered a relative error of modeling, and it can be used as a standard to verify the program performance.

When the namelist variable *lmbudg* is set to true, the mass budgets are written to stand-alone output data files. There will be an output data file for each grid nest at every output time.

All of the items above, except item 1, are expressed as a percentage of the total mass production (1). This allows for easy estimation of the orders of aerosol-tracer production magnitude. Any abnormal values in the mass budgets should indicate where the program is functioning inappropriately. Therefore, the mass budget becomes a useful tool in testing numerical schemes and in model development, modification, and upgrading.

5.9 Examples of COAMPS Aerosol-Tracer Simulations

5.9.1 April 1998 Asian dust storms

Every year during the spring when weather is dry and strong winds are present, the China and Mongolian deserts of Taklamakan, Gobi, Ordos, and Loesslands produce great amounts of dust. In mid-April 1998, the Gobi and

Taklamakan deserts were sources of two large dust events and numerous smaller events (Figure 17 (bottom panel)). The dust observations, obtained from surface weather stations in the source area, were taken 14–23 April. The time series shows that the first strong event occurred on 15 April. Subsequent peaks occurred on 19 and 20 April. The dust plume of the 19th traversed over the Pacific Ocean and reached North America one week later (Husar et al. 1998).

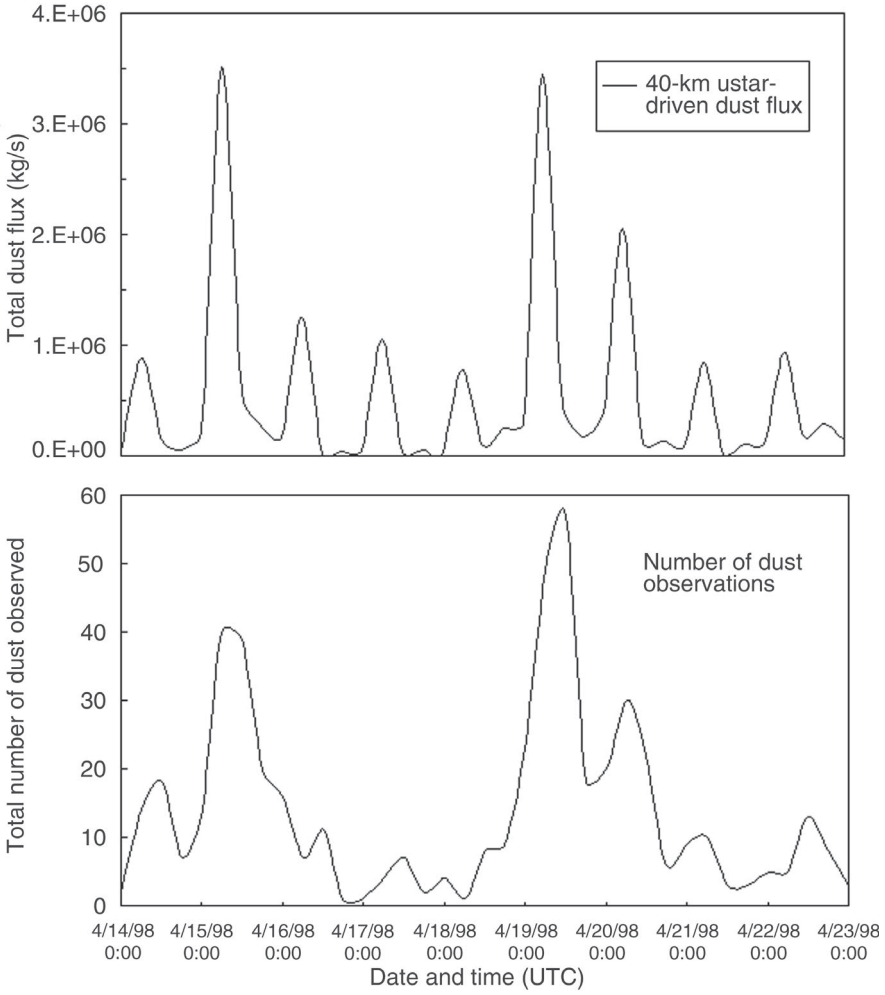


Figure 17.
Total number of surface dust observations from weather stations compared with total instantaneous dust fluxes of model 40-km resolution in the dust source areas at 0000, 0600, 1200, and 1800 UTC.

The total dust fluxes produced by COAMPS are integrated over the same source area and compared with the total number of dust observations for the same period (Figure 17, top panel). The COAMPS simulation captures these three major events very well and shows dust production occurring on other days as well. The modeled dust fluxes generally peak during daytime, while the minimums occur during nighttime. This indicates that COAMPS is able to predict diurnal variation in dust production.

The simulation is made with a 40-km grid resolution, and the dust production calculated by the u_*-driven flux formula (Equation (5.17)). The dust is modeled with 20 bins, ranging from 0.01 to 40.0 μm in particle radius and in lognormal size distribution. It includes major dust physics of sedimentation, dry deposition, and wet removal. The simulation spans 10–24 April in a large domain covering Mongolia, China, and neighboring areas. Figure 18 shows the modeled dust plume (represented by the mass load field, e.g., vertical integral of concentration) on 19 April. The plume is a combined result of dust generation and dust downstream transport occurring at the same time. Figure 19 shows the observed dust plume obtained from AVHRR satellite images on 19 April. The location and the shape of the modeled dust plumes are consistent with the satellite observation.

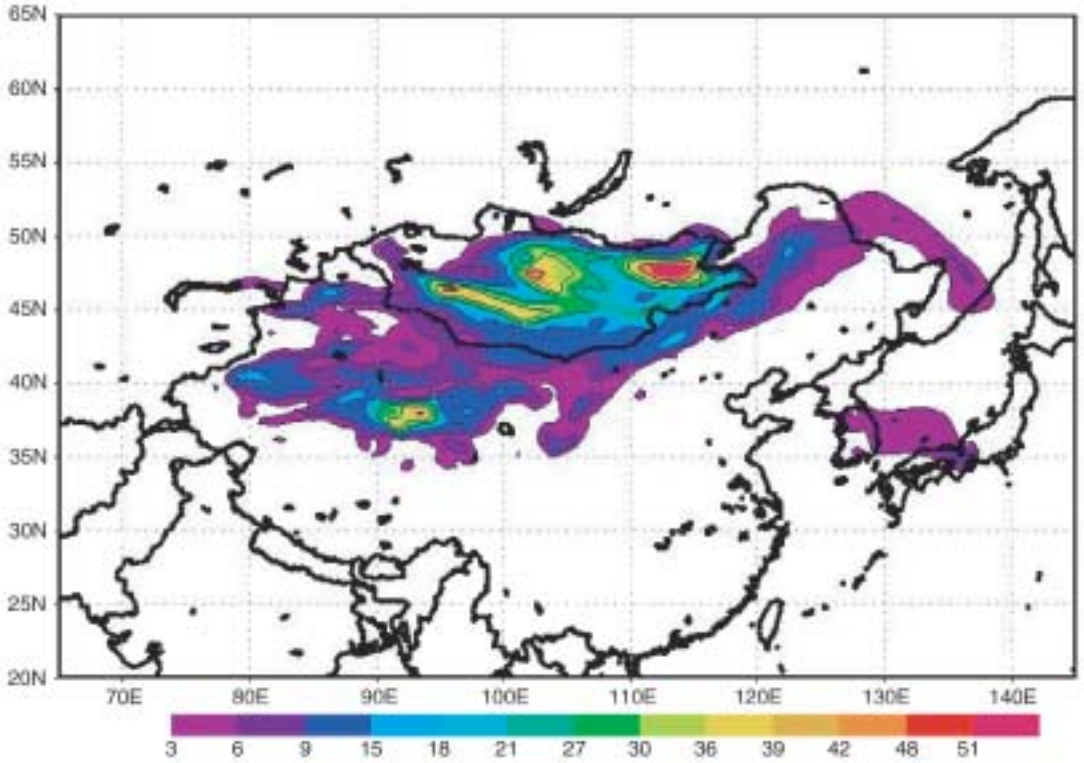


Figure 18.
Modeled dust storms in mass load (g m^{-2}) of 40-km resolution at 1200 UTC, 19 April 1998 in Mongolia and China.

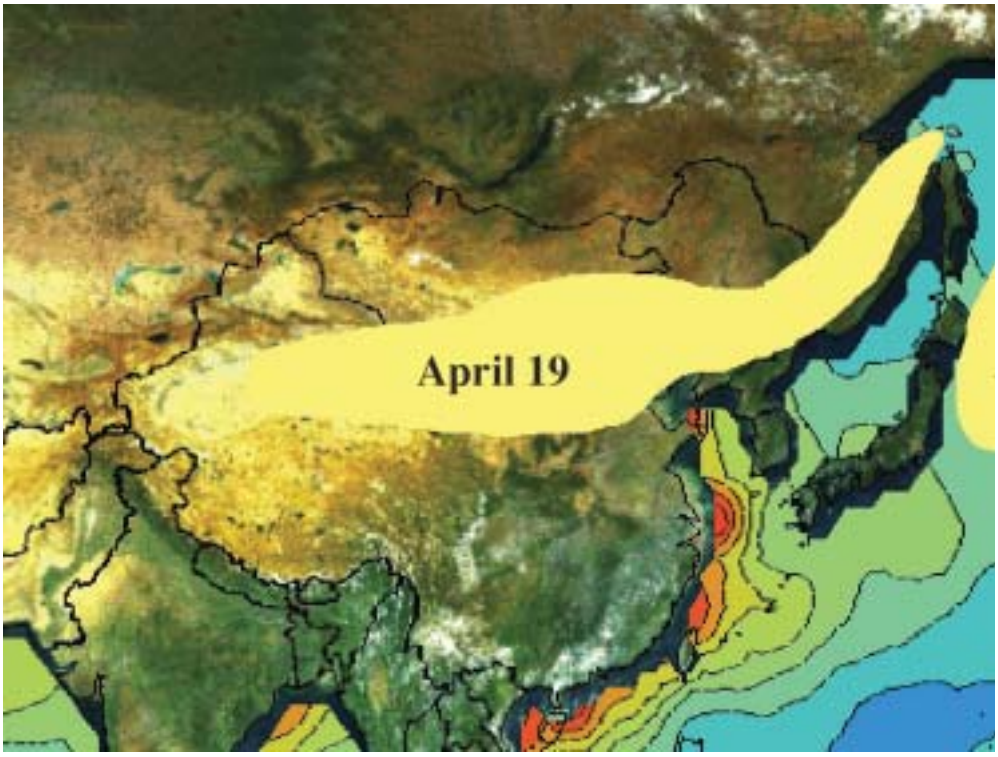


Figure 19.
East Asian dust plume on 19 April 1998, derived from satellite images and TOMS aerosol index data by Husar et al. (2001).

5.9.2 July 2001 Mt. Etna volcano ash plumes

Mt. Etna, Europe's most active volcano, is located near the east coast of Sicily, Italy. It consists of several vents standing about 3,300 m tall, with a broad base spanning roughly 60 × 40 km. On 18 July 2001 and subsequent days, Mt. Etna erupted. The volcano released thick lava flows and vast columns of steam and smoke (ash). Satellite pictures show brown ash plumes hovering over the Mediterranean Sea (Figure 20). The plumes originated from the volcano on 22 and 24 July. More occurred several days later.

To simulate this volcano ash plume event, Mt. Etna is taken as a point source of continuous emission, with emission strength of 100 kg/s in a one-day period for each plume episode. The emission height is 600 m above the top of the volcano. For the ash plume observed on 24 April (Figure 20), the emission in the model was set to begin at 2300Z on the 23 April with ash considered as a passive tracer.

Figure 20.
Satellite image of Mt. Etna volcano ash plume at 1200 UTC 24 July 2001 over the Mediterranean Sea.

The purpose of this simulation is to demonstrate the importance and practical usage of the nesting interaction described in Section 4.6. Three nested grids were used: a 27-km domain providing the large-scale background dynamic forcing, a 9-km domain containing the whole range of ash plume transport, and a 3-km domain covering only the emission source area.

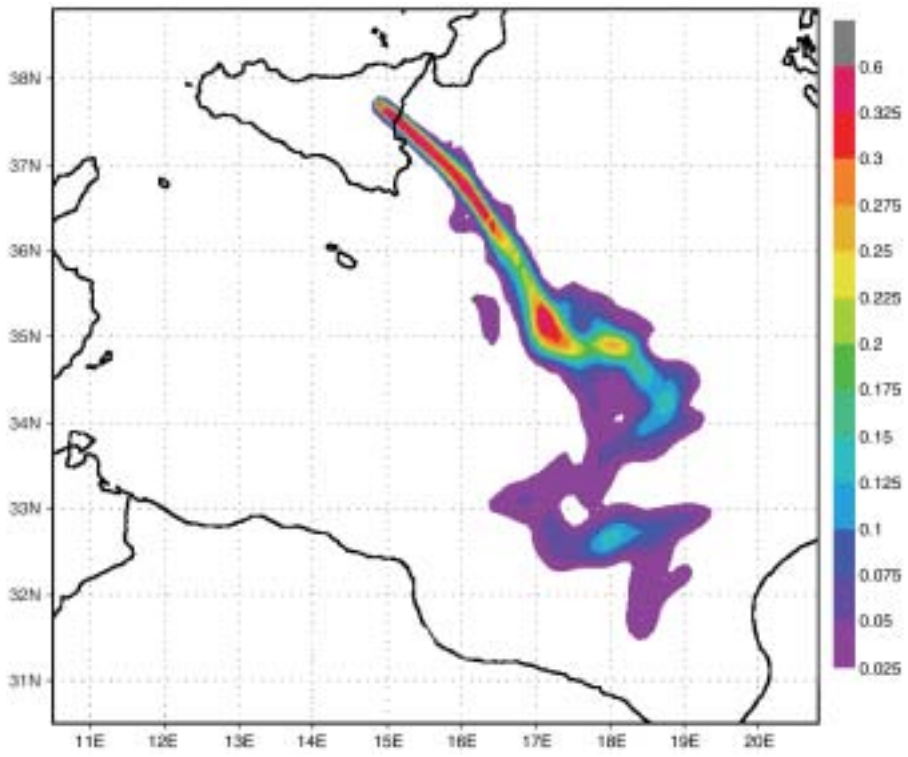


Figure 21.
Modeled Mt. Etna volcano ash plume in mass load (g m^{-2}) at 1200Z 24 July 2001, in 9-km resolution obtained with grid-nest interaction from 3-km subdomain of source area.

Two model simulations are performed from 20–25 July. The first model run has the two-way nested grid interaction. The second run is the same as the first run except without the two-way interaction. Figures 21 and 22 show the modeled ash plume (mass load field) at 1200Z of 24 April from the first and the second model simulation. Both modeled plumes exhibit similar features of transport and downstream meandering in comparison with the satellite image (Figure 20). With two-way nested grid interaction (e.g., the feedback from high-resolution to low resolution grids), the upstream plume from the 9-km resolution is confined with the 3-km mass concentration field in Figure 21. It also appears to be narrower than the one in Figure 22, which has no feedback from the 3-km grid.

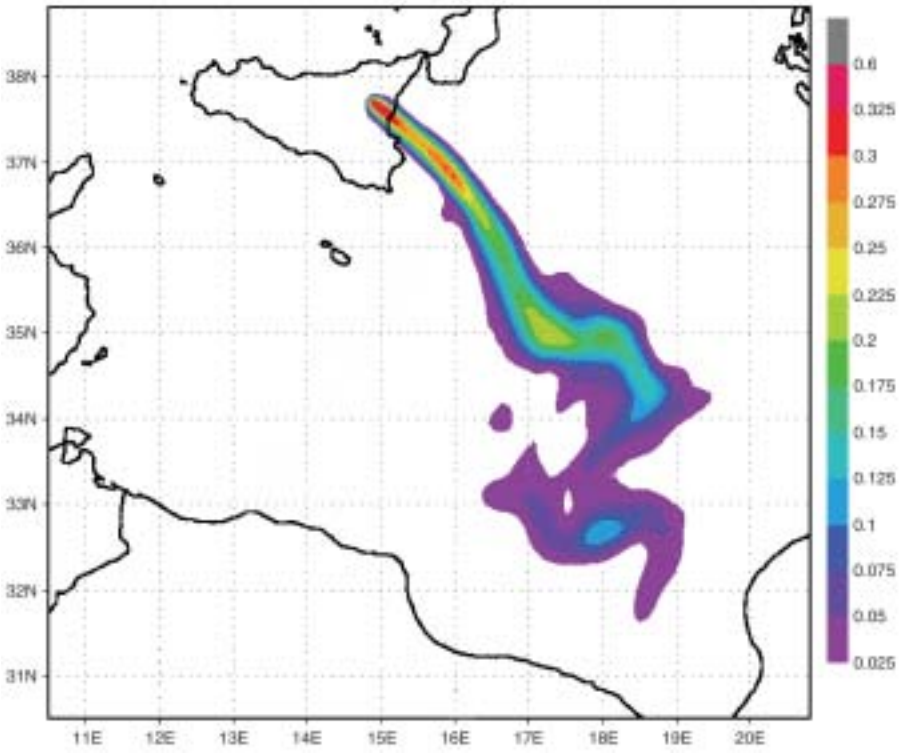


Figure 22.
Modeled Mt. Etna volcano ash plume in mass load (g m^{-2}) at 1200Z 24 July 2001, in 9-km resolution without grid-nest interaction.

REFERENCES

Alfarao, S.C., A. Gaudichet, L. Gomes, and M. Maille, 1997: Modeling the size distribution of a soil aerosol produced by sandblasting. *J. Geophys. Res.*, **102**, 11239-11249.

Arakawa, A. and V.R. Lamb, 1977: Computational design of the basic dynamical processes of the UCLA general circulation model. *Methods in Computational Physics*, 17, Academic Press, 173-265.

Asai, T., 1965: A numerical study of the air-mass transformation over the Japan Sea in winter. *J. Met. Soc. Japan*, **43**, 1-15.

Asselin, R.A., 1972: Frequency filter for time integrations. *Mon. Wea. Rev*., **100**, 487-490.

Barker, E., 1992: Design of the Navy's Multivariate Optimum Interpolation Analysis System. *Weather and Forecasting*, **7**, 220-231.

Bennett, A., 1992: Inverse methods in physical oceanography. Cambridge University Press, New York, 346 pp.

Blackadar, A.K., 1962: The vertical distribution of wind and turbulent exchange in a neutral atmosphere. *J. Geophys. Res.*, **67**, 3095-3102.

Bott, A., 1989a: A positive definite advection scheme obtained by nonlinear renormalization of the advective fluxes. *Mon. Wea. Rev.*, **117**, 1006-1015.

Bott, A., 1989b: Reply. *Mon. Wea. Rev.*, **117**, 2633-2636.

Bott, A., 1992: Monotone flux limitation in the area-preserving flux-form advection algorithm. *Mon. Wea. Rev.*, **120**, 2592-2602.

Carmichael, G.R., L.K. Peters, and T. Kitada, 1986: A second generation model for regional-scale transport/chemistry/deposition. *Atmos. Environ.*, **20**, 173-188.

Chelton, D.B., R.A. DeSzoeke, M.G. Schlax, K.E. Naggar, and N. Siwertz, 1998: Geographical variability of the first baroclinic Rossby radius of deformation. J. *Physical Oceanogr*. **28**:433-460.

Christensen, J.H., 1997: The Danish Eulerian hemispheric model – a three-dimensional air pollution model used for the arctic. *Atmos. Environ.*, **31**, 4169-4191.

Clements, T., T.H. Merriam, R. O. Stone, J. L. Eimann, and H. L. Reade, 1957: A study of desert surface conditions, Tech. Rep. EP-53, Quartermaster Research and Development Command, U. S. Army, Natick Laboratories, Natick, Massachusetts, 130pp. [NTIS PB140373].

Cooper, M. and K.A. Haines, 1996: Altimetric assimilation with water property conservation. *J. Geophys. Res*. 24:1059-1077.

Cotton, W.R., G.J. Tripoli, R.M. Rauber, and E.A. Mulvihill, 1985: Numerical simulation of the effects of varying ice crystal nucleation rates and aggregation processes on orographic snowfall. *J. Climate Appl. Meteor.,* **25**, 1658-1680.

Cotton, W.R. and R.A. Anthes, 1989: Storm and cloud dynamics. Academic Press, Inc., 883 pp.

Daley, R., 1991: Atmospheric Data Analysis. Cambridge University Press, Cambridge, 457 pp.

Davies, R., 1982: Documentation of the solar radiation parameterization in the GLAS climate model. NASA Technical Memorandum 83961, 57pp.

Fairall, C.W., E.F. Bradley, D.P. Rogers, J.B. Edson, and G.S. Young, 1996: Bulk parameterization of air-sea fluxes for tropical ocean-global atmospheric coupled-ocean atmospheric response experiment. *J. Geophys. Res.*, **101**, 3747-3764.

Fletcher, N.H., 1962: *The Physics of Rainclouds.* Cambridge University Press, 386 pp.

Fofonoff, N.P. and R.C. Millard, 1983: Algorithms for computation of fundamental properties of seawater. *Tech. Pap. Mar. Sci*. UNESCO **44**, 53 pp.

Fox, D.N., W.J. Teague, C.N. Barron, M.R. Carnes, and C.M. Lee, 2002: The Modular Ocean Data Assimilation System. *J. Atmos. Ocean. Technol*. **19**:240-252.

Fritsch, J.M. and C.F. Chappell, 1980: Numerical prediction of convectively driven mesoscale pressure systems. Part I: Convective parameterization. *J. Atmos. Sci.*, **37**, 1722-1733.

Fujita, T., 1959: Precipitation and cold air production in mesoscale thunderstorm systems. *J. Meteor.*, **16,** 454-466.

Gal-Chen, T. and R.C.J. Somerville, 1975: On the use of a coordinate transformation for the solution of the Navier-Stokes equations. *J. Comput. Phys.*, **17**, 209-228.

Gillette, D.A., 1978: A wind tunnel simulation of the erosion of soil: Effect of soil texture, sandblasting, wind speed, and soil consolidation on dust production, *Atmos. Environ*., **12**, 1735-1743.

Gillette, D.A. and R. Passi, 1988: Modeling dust emission caused by wind erosion. *J. Geophys. Res.-atmos*., **93**, D11, 14233-14242.

Goerss, J. and P. Phoebus, 1992: The Navy's Operational Atmospheric Analysis. *Weather and Forecasting,* **7**, 232-249.

Haack, T., 1996: Software User's Manual for the Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS™), Naval Research Laboratory Technical Report, NRL/MR/7543-96-7227, 88 pp.

Hallet J. and S.C. Mossop, 1974: Production of secondary ice particles during the riming process. *Nature* (London), **249**, 26-28.

Haltiner, G.J. and F.L. Martin, 1957: *Dynamical and Physical Meteorology*. McGraw-Hill, New York, NY. 470 pp.

Haltiner, G.J. and R.T. Williams, 1980: Numerical weather prediction and dynamic meteorology. John Wiley and Sons, 477 pp.

Harshvardhan, R. Davies, D. Randall, and T. Corsetti, 1987: A fast radiation parameterization for atmospheric circulation models. *J. Geophys. Res*., **92**, 1009-1016.

Hogan, T.F. and T.E. Rosmond, 1991: The description of the U.S. Navy Operational Global Atmospheric Prediction System's spectral forecast model. *Mon. Wea. Rev*., **119**, 1786-1815.

Hollingsworth, A. and P. Lonnberg, 1986: The statistical structure of short-range forecast errors as determined from radiosonde data. Part I: The windfield. *Tellus* **38A**, 111-136.

Hobbs, P.V. and A.L. Rangno, 1985: Ice particle concentrations in clouds. *J. Atmos. Sci*., **42**, 2523-2549.

Hodur, R.M., 1997: The Naval Research Laboratory's Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS). *Mon. Wea. Rev*., **135**, 1414-1430.

Husar, R.B. et al., 2001: The Asian dust events of April 1998. *J. Geophys. Res.-atmos.*, **106**, D16.

Joseph, J.H., W.J. Wiscombe, and J.A. Weinman, 1976: The delta-Eddington approximation for radiation flux transfer. *J. Atmos. Sci*., **33**, 2452-2459.

Kain, J.S. and J.M. Fritsch, 1990: A one-dimensional entraining/detraining plume model and its application in convective parameterization. *J. Atmos. Sci.,* **47,** 2784-2802.

Kain, J.S. and J.M. Fritsch, 1993: Convective parameterization for mesoscale models: The Kain-Fritsch scheme. *The Representation of Cumulus Convection in Numerical Models, Meteor. Monogr.,* No. 46. Amer. Meteor. Soc., 165-170.

Kessler, E., III, 1969: On the distribution and continuity of water substance in atmospheric circulations. *Meteor. Monogr.,* No. 32, Amer. Meteor. Soc., 84 pp.

Khairoutdinov, M. and Y. Kogan, 2000: A new cloud physics parameterization in a large-eddy simulation model of marine stratocumulus. *Mon. Wea. Rev*., **128**, 229-243.

Klemp, J. and R. Wilhelmson, 1978: The simulation of three-dimensional convective storm dynamics. *J. Atmos. Sci.*, **35**, 1070-1096.

Lacis, A.A. and J.E. Hansen, 1974: A parameterization for the absorption of solar radiation in the earth's atmosphere. *J. Atmos. Sci*., **31**, 118-133.

Liu, M. and J.J. Carroll, 1996: A high-resolution air pollution model suitable for dispersion studies in complex terrain. *Mon. Wea. Rev*., **124**, 2396-2409.

Liu, M. and D.L. Westphal, 2001: A study of the sensitivity of simulated mineral dust production to model resolution. *J. Geophys. Res.-atmos*., **106**, D16, 18099-18112.

Lin. Y.-L., R.D. Farley., and H.D. Orville, 1983: Bulk parameterization of the snow field in a cloud model. *J. Climate Appl. Meteor.,* **22**, 1065-1092.

Lonnberg, P. and A. Hollingsworth, 1986: The statistical structure of short-range forecast errors as determined from radiosonde data. Part II: The covariance of height and wind errors. *Tellus* **38A**:137-161.

Lorenc, A., 1981: A global three-dimensional multivariate statistical analysis system. *Mon. Wea. Rev*. **109**:701-721.

Louis, J.-F., 1979: A parametric model of vertical eddy fluxes in the atmosphere. *Bound. Layer. Meteor.,* **17**, 187-202.

Marshall, J.S. and W.M. Palmer, 1948: The distribution of raindrops with size. *J. Meteor.,* **5**, 165-166.

Marticorena, B. and G. Bergametti, 1995: Modeling the atmospheric dust cycle: 1. Design of a soil-derived dust emission scheme. *J. Geophys. Res.-atmos*., **100**, D8, 16415-16430.

Mellor, G.L. and T. Yamada, 1974: Development of a turbulence closure for geophysical fluid problems. *Rev. Geophys. and Space Phys*., **20**, 851-875.

Meyers, M.P., P.J. DeMott, and W.R. Cotton, 1992: New primary ice-nucleation parameterizations in an explicit cloud model. *J. Applied. Meteor*., **31**, 708-721.

Meyers, M.P., R.L. Walko, J.Y. Harrington, and W.R. Cotton, 1997: New RAMS cloud microphysics paramterization. Part II: The two-moment scheme. *Atmos. Res*., **45**, 3-39.

Nickling, W.G. and J.A. Gillies, 1993: Dust emission and transport in Mali, West Africa. *Sedimentology,* **40**, 859-868.

Ogura, Y. and H.-R. Cho, 1973: Diagnostic determination of cumulus cloud populations from observed large-scale variables. *J. Atmos. Sci.,* **30,** 1276-1286.

Pruppacher, H.R. and J.D. Klett, 1978: *Microphysics of Clouds and Precipitation*. D. Reidel, 398 pp.

Raymond, D.J. and A.M. Blyth, 1986: A stochastic mixing model for nonprecipitating cumulus clouds. *J. Atmos. Sci.,* **43,** 2708-2718.

Reisner, J., R.M. Rasmussen, and R.T. Bruintjes, 1998: Explicit forecasting of supercooled liquid water in winter storms using the MM5 mesoscale model. *Q.J.R. Meteor. Soc*., **124**, 1071-1107.

Riishojgaard, L., 1998: A direct way of specifying flow dependent background error for meteorological analysis systems. *Tellus* **50A**: 42-57.

Robert, A.J., 1966: The investigation of a low-order spectral form of the primitive equation models. *J. Meteor. Soc. Japan.*, **44**, 237-245.

Rutledge, S.A. and P.V. Hobbs, 1983: The mesoscale and microscale structure and organization of clouds and precipitation in midlatitude cyclones. VIII: A model for the “seeder-feeder” process in warm-frontal rainbands. *J. Atmos. Sci*., **40**, 1185-1206.

Rutledge, S.A. and P.V. Hobbs, 1984: The mesoscale and microscale structure and organization of clouds and precipitation in midlatitude cyclones. XII: A diagnostic modeling study of precipitation development in narrow cold-frontal rainbands. *J. Atmos. Sci*., **41**, 2949-2972.

Sagan, C. and J.B. Pollack, 1967: Anisotropic nonconservative scattering and the clouds of Venus. *J. Geophys. Res*., **72**, 466-477.

Seinfeld, J.H., 1986: *Atmospheric Chemistry and Physics of Air Pollution*. John Wiley and Sons, 307-338 pp.

Skamarock, W.C. and J.B. Klemp, 1992: The Stability of Time-Split Numerical Methods for the Hydrostatic and the Nonhydrostatic Elastic Equations. *Mon. Wea. Rev.,* **120**, 2109–2127.

Slinn, A.A. and W.G.N. Slinn, 1980: Predictions for particle deposition on natural waters.–*Atmos. Environ.,* **14**, 1013-1016.

Soong, S. and Y. Ogura, 1973: A comparison between axisymmetric and slab-symmetric cumulus cloud models. *J. Atmos. Sci*., **30**, 879-893.

Stull, R.B., 1988: *An Introduction to Boundary Layer Meteorology*. Kluwer Academic Publishers, Dordrecht, the Netherlands. 251-289

Tao, W., J. Simpson, and M. McCumber, 1989: An ice-water saturation adjustment. *Mon. Wea. Rev.*, **117**, 231-235.

Therry, G. and P. Lacarrere, 1983: Improving the kinetic energy model for planetary boundary layer description. *Bound. Layer Meteor.,* **25**, 63-88.

Toon, O.B., R.P. Turco, D. Westphal, R. Malocone, and M.S. Liu, 1988: A multidimensional model for aerosols: description of computational analogs. *J. Atmos. Sci.*, **45**, 2123-2143.

Tremback, C.J., J. Powell, W.R. Cotton, and R.A. Pielke, 1987: The forward-in-time upstream advection scheme: Extension to higher orders. *Mon. Wea. Rev.,* **115**, 540-555.

Tripoli, G.J. and W.R. Cotton, 1982: The Colorado State University three-dimensional cloud/mesoscale model – 1982. Part I: General theoretical framework and sensitivity experiments. *J. Rech. Atmos*., **16**, 185-220.

Tripoli, G.J., 1992a: A nonhydrostatic mesoscale model designed to simulate scale interaction. *Mon. Wea. Rev.*, **120**, 1342-1359.

Tripoli, G.J., 1992b: An explicit three-dimensional nonhydrostatic numerical simulation of a tropical cyclone. *Meteor. Atmos. Phys*., **49**, 229-254.

Walcek, C.J., R.A. Brost, J.S. Chang, and M.L. Wesely, 1986: SO_2, sulfate and HNO_3 deposition velocities computed using regional landuse and meteorological data. *Atmos. Environ.*, **20**, 949-964.

Weisman, M.L., W.C. Skamarock, and J.B. Klemp, 1997: The resolution dependence of explicitly modeled convective systems. *Mon. Wea. Rev*., **125**, 527-548.

Westphal, D.L., O.B. Toon, and T.N. Carlson, 1987: A two-dimensional numerical investigation of the dynamics and microphysics of Saharan dust storms. *J. Geophys. Res.*, **92**, 3027-3049.

Westphal, D.L., O.B. Toon, and T.N. Carlson, 1988: A case study of transport and mobilization of Saharan dust. *J. Atmos. Sci.*, **45**, 2145-2175.

Yamada, T., 1983: Simulations of nocturnal drainage flows by a q^2l turbulence closure model. *J. Atmos. Sci*., **40**, 91-106.

Zhang, D.–L., 1989, and J.M. Fritsch, 1986: Numerical simulation of the meso-β scale structure and evolution of the 1977 Johnstown flood. Part I: Model description and verification. *J. Atmos. Sci.,* **43,** 1913-1943.

Appendix A

GLOSSARY

Aerosol Tracer

VARIABLES:

aerosol/tracer sink (C_{snk})
aerosol/tracer source (C_{src})
air density (ρ_a)
cloud depth in meters (H using washout)
constant coefficient (c)
Cunningham correction factor (C_c)
deposition velocity (v_d)
dimensional constant (A)
extinction geometry factor (Q)
fraction of the model grid box that is dust erodible (f)
free parameter (ν)
geometric standard deviation of the jth mode (σ_j)
gravitational fall velocity (v_f)
gravity (g)
horizontal eddy diffusivity (K_x and K_y)
horizontal velocity components (u and v)
indexed mode (j)
kinetic viscosity of air (ν)
layer thickness (Δz_k)
mass concentration (C)
mass flux caused by deposition (F_{dep})
mass fraction of particles for mode j (M_j)
mass generated from sources (M)
mass median radius (R_{mj})
Monin-Obukhov length (L)
number of model vertical levels (n)
number of particles in a unit volume (N)
optical thickness (τ_i)
parameterized scavenging rate (Λ)
particle density (ρ_p)
particle mass (M_p)
particle radius (R)
particle radius (R_i)
particle radius (R_p)
ratio of adjacent size bins of particles (V_{rat})
representative particle radius of the ith bin (R_i)
size bin (i)
subgrid scale turbulent mixing (D_x, D_y, and D_σ)
terminal velocity (v_f)
terrain coordinate (σ)
terrain height (z_{sfc})

threshold friction velocity (u_{*_t})
vertical altitude (z_{top})
vertical diffusion (K_z)
vertical dust flux (F)
vertical grid layer size in meters (H using rainout)
wind speed at a 10-meter height (U_{10})

Convective Scheme

VARIABLES:
acceleration due to gravity (g)
change in pressure over time (ω)
change in total condensate in a layer due to precipitation (δr_c)
cloud water (q_c)
contributions to ω from downdraft (ω_d)
contributions to ω from environmental subsidence ($\tilde{\omega}$)
contributions to ω from updraft (ω_u)
Exner's function (π)
horizontal area of a model grid cell (A)
latent heat released from changes in q_v due to change of phase (L)
layer-mean vertical velocity (w)
potential temperature (θ)
pressure (p)
rate constant (c_1)
rate of detrainment of updraft and downdraft mass (δ_u and δ_d)
rate of entrainment of updraft and downdraft environmental mass (ε_u and ε_d)
sum of condensate concentration at the bottom of the layer and half the degree of supersaturation at the top (r_{c0})
thickness (δz)
total amount of condensate in the updraft (q_l)
updraft mass flux (M_u)
water vapor mixing ratio (q_v)

Radiation

VARIABLES:
absorbed shortwave flux (A)
absorption coefficient (k)
Boltzmann's constant (K)
composite reflection coefficient ($\hat{R}_k$)
Delta-Eddington parameter (χ_k)
diffuse transmission between levels a and b (Γ_a^b)
diffuse transmission coefficient (t_k)
direct solar flux (S_k)
downward diffuse flux of the two-stream solution ($\tilde{D}_k$)
downward flux of the adding solution ($\hat{D}_k$)
downward longwave flux ($F^{\downarrow}$)
effective path (x)
equivalent absorption coefficient (κ)
frequency (v)
index for air surface (s)
index for earth ground (g)
index for model top (p_{top})
index for vertical level (k)

magnification factor ($\hat{M}_k$)
magnification factor (M)
ozone level (k)
ozone optical depth (τ)
Planck's constant (h)
probability distribution (p)
probability of a clear line of sight (P)
probability of a clear line of sight between levels a and b (P_a^b)
reflected radiation effective path (x^*)
reflection (R)
reflection coefficient (γ_k)
sine of the solar angle (μ_0)
single scattering albedo (ω_{0k})
solar angle (θ_0)
solar constant (S_0)
speed of light (c)
total albedo of the underlying atmosphere to visible and ultraviolet light (R_{oz})
total downward flux (D_k)
total number of vertical levels (N)
total optical depth (τ_k^*)
total reflection coefficient ($\hat{\Gamma}_k$)
total upward flux (U_k)
transmission (T)
transmission function (Γ)
updaward diffuse flux of the two-stream solution ($\tilde{U}_k$)
upward flux of the adding solution ($\hat{U}_k$)
upward longwave flux ($F^{\uparrow}$)

TKE Scheme

VARIABLES:
acceleration due to gravity (g)
air-sea temperature difference ($\Delta\theta$)
Charnock constant (c_0)
coefficient of thermal expansion (β)
constant (c_υ)
constant (S_e)
constant dependent upon stability (α)
dissipation length scale (Λ_1)
eddy coefficients ($K_{H,M,e}$)
elevation (z)
friction velocity (u_*)
horizontal velocity field (U)
horizontal velocity field (V)
master length scale (l)
mean potential temperature over the depth of the surface layer (Θ)
molecular viscosity (υ)
neutral drag coefficient (a^2)
polynomial functions of the flux Richardson Number ($S_{H,M}$)
polynomial functions to compute eddy coefficients (Ri_f, S_H, and S_M)
potential temperature (θ)
ratio of the transfer coefficient for heat to that for momentum (R)

surface roughness (z_0)
three-dimensional turbulent velocity field (u', v', and w')
virtual potential temperature (θ_v)
von Karaman constant (κ)
wind speed at reference elevation (u)

Appendix B

NAMELIST TABLES

***atmosnl* Namelist**

Parameter	Description	Default Value
cdtg	Date-time-group (character*10)	1995010100
iaero	Passive-tracer switch 0 = off 1 = on	0
ianal	Analysis boundary condition switch 0 = compute output boundary tendencies for the coarse domain only 1 = compute analysis and boundary tendencies 2 = compute analysis only 3 = compute Nowcast only	1
idelay	Delay time for nests (hour, minute, second for each nest)	21*3
ldigit	Digital filter switch (logical)	false
lgrdall	Passive tracer grid switch (logical) true = passive tracer for all grids false = passive tracer for specified grids only	true
lnmove	Switch to move nests (logical) true = move nests false = no move	*mxgrds**false
lm	Number of atmospheric analysis pressure levels	16
lmbc	Number of atmospheric boundary condition pressure levels	16
maxnest	Maximum number of nests	*mxgrds*
mbin	Number of passive tracer bins	1
mspc	Number of passive tracer species	1
nbdya	Boundary condition option used for digital filter	7
nbdypt	Number of grid points used for boundary condition blending	5
nestcc	Grid number for passive tracer option *lgrdall* = false	1
nbnam	Number of halo points	1
loopvecnam	Loop vectorization option 0 = no vectorization 1 = vectorization	1
ndxnam	Number of domains in x-direction for each nest	1,1,1,1,1,1,1
ndynam	Number of domains in y-direction for each nest	1,1,1,1,1,1,1
npr0nam	Processor number for root processor *npr0*	0

***coamnl* Namelist**

Parameter	Description	Default Value
ddata	Path for idealized data (char*80)	‘ ‘
messg	Message (char*32)	‘ ‘
type1k	High-resolution terrain data type (char*32)	DTED1_010
cstline	Coastline parameter (real)	*maxgrds**0.0
hmt	Idealized mountain height (real)	0.0
iwvlng	Silhouette terrain parameter (real)	2.0
nslflt	Silhouette terrain filter parameter	*maxgrds**0
nsrch	Search box size for silhouette terrain	2,2,5,2,1,1,1
prbc	Pressure levels (hPa) for analysis boundary conditions (real)	10.0, 20.0, 30.0, 50.0, 70.0, 100.0, 150.0 200.0, 250.0, 300.0, 400.0, 500.0, 700.0 850.0, 925.0,1000.0,84*0.0
r1dlat	Land-water latitude location parameter (real)	*ilen**0.0
r1dlon	Land-water longitude location parameter (real)	*ilen**0.0
silwgt	Silhouette terrain weight parameter (real)	*maxgrds**1.0
silmax	Silhouette terrain averaging parameter (real)	*maxgrds**1.0
toffset	Idealized offset value (real)	0.0
topores	Terrain horizontal resolution (km) (real)	*maxgrds**1.0
ibiod	Output parameter for surface fields	0
ibog1	Size of pseudo-observations analysis volumes (x-direction)	0
icup	Cumulus parameterization type	1,1,1,1,1,1,1
idbms	I/O parameter type	1
ilandu	Land-use type	*maxgrds**0
ilw1d	Land-water parameter	*ilen**9
ioizi	Zero-increment option for MVOI	*maxgrds**0
iraob1	Threshold number of rawindsond for each MVOI volume	2
meso_init	Option for interpolating first guess fields to COAMPS coarse mesh 0 = use NOGAPS as first guess and boundary condition 1 = use COAMPS as first guess and boundary condition 2 = output additional COAMPS flat files to be used as boundary condition for the next meso-init forecast.	0
meso_grid	Grid number for meso_init	1
meso_start	Start time for meso_init	0,0,0
meso_end	End time for meso_init	0,0,0
meso_cycle	Cycle time for meso_init	0,0,0
lm_meso	Number of vertical levels for meso_init	30

***coamnl* Namelist (continued)**

Parameter	Description	Default Value
pr_meso	Pressure levels (hPa) for meso_init (real)	10.0, 20.0, 30.0, 50.0, 70.0, 100.0, 150.0 200.0, 250.0, 300.0, 350.0, 400.0,450.0 500.0 ,550.0, 600.0, 650.0, 700.0, 750.0 775.0, 800.0, 825.0 ,850.0, 875.0, 900.0, 925.0, 950.0, 975.0, 1000.0, 1013.0,70*0.0
isoiflg	Idealized deep soil temperature flag	*maxgrds**1
itaubc	Starting TAU to search for NOGAPS boundary condition	0
itaung	Time interval to search for NOGAPS boundary condition	6
itaus	Starting time for analysis boundary condition	0,0,0
i1km	High resolution terrain index	1
jbog1	Size of pseudo-observations analysis volumes (y-direction)	0
jcm2fg	Idealized interpolation parameter	0
ksavdig	Digital filter time parameter	-1,0,0
digtim	Digital filter time parameter (real)	6.0, 0.,0
ksavef	Restart save parameter	-1,0,0
ksaver	Restart save parameter	-1,0,0
ksavpcp	Precipitation bucket frequency parameter	*maxgrds**(72,0,0)
ksavmsp	Maximum wind speed output parameter	-1,0,0
nbio	Surface output field number parameter	1
nbiox	Surface output field number parameter	*maxpt1d**5
nbioy	Surface output field number parameter	*maxpt1d**5
noiexp	Number of MVOI expansions	3
ifsave	Output save parameter	1
isavefrq	Output save frequency parameter	-1,0,0
lshrad	Number of shortwave radiation passes	4
ncast_start	Nowcast start parameter	0
ncast_end	Nowcast end parameter	0
ncast_cycle	Nowcast cycle parameter	0
kmbbeg	Budget begin parameter	99999
kmbend	Budget end parameter	99999
kmbinc	Budget increment parameter	99999
imovtyp	Moving nest type parameter	*maxgrds**1
itmove	Moving nest move times	*ilen3**-1
nmovei	Number of move grid points in x-direction	*ilen**0
nmovej	Number of move grid points in y-direction	*ilen**0
movemx	Maximum number of move grid points per time step	1,3,9,27,27,27,27
xlatmov	Ending latitude for *imovtyp* = 2	*ilen**-999.99
xlonmov	Ending longitude for *imovtyp* = 2	*ilen**-999.99
ioffset	Idealized data offset parameter	0
joffset	Idealized data offset parameter	0
kwidth	Idealized width parameter	0
nwidth	Idealized width parameter	0

***coamnl* Namelist (continued)**

Parameter	Description	Default Value
lcoare	TOGA COARE surface flux parameter (logical)	false
lcupar	Cumulus parameterization parameter (logical)	true
lpseud	Pseudo-observation parameter (logical)	false
lsilhout	Silhouette terrain parameter (logical)	false
lvisfn	Output visual parameter (logical)	false
lviz5d	Output VIS5D parameter (logical)	false
l1dij	One-dimensional output parameter (logical)	true
l2way	Two-way interaction parameter (logical)	false
l2wayi	Two-way interaction parameter (logical)	true
labdye	Parameter to turn on/off *abdye* (logical)	true
labdytq	Parameter to turn on/off *abdytq* (logical)	true
labdyuv	Parameter to turn on/off *abdyuv* (logical)	true
lafore	Parameter to turn on/off *afore* (logical)	true
laforqx	Parameter to turn on/off *aforqx* (logical)	true
lalhs	Parameter to turn on/off *alhs* (logical)	true
lamixtq	Parameter to turn on/off *amixtq* (logical)	true
lamixnf	Parameter to turn on/off *amixnf* (logical)	true
lamixuv	Parameter to turn on/off *amixuv* (logical)	true
lamixw	Parameter to turn on/off *amixw* (logical)	true
larhsp	Parameter to turn on/off *arhsp* (logical)	true
larhsu	Parameter to turn on/off *arhsu* (logical)	true
larhsv	Parameter to turn on/off *arhsv* (logical)	true
larhsw	Parameter to turn on/off *arhsw* (logical)	true
lnestf	Halo point communication parameter (logical)	true
ldiagt	Diagnostic print parameter (logical)	false
lmpi2	MPI-1/MPI-2 parameter (logical) false = MPI-1 true = MPI-2	false

***coamnl* Data Path Namelist Variables**

Type of Model Run (Namelist variables)	Directory Locations of Initial Input Data (Namelist variables)
operational or research and development (*ldbms*, *ldbmi*, *ldbmo*)	surface databases (*dsclim*, *dsgiss*, *dsdted*, *dsland*, *dslanu*)
idealized or real data (*icase*)	observational data (*datfil*)
data assimilation update (*iupd*, *loi*, *loimf*)	NOGAPS fields (*dsetng*, *dsngff*) COAMPS fields (*dsetg*, *dsnrff*)

COAMPS Ocean Analysis *oanl* Namelist

Name	Type	Description	Values
blist	char	black listed call signs	no default call signs
case	integer	option for creating initial conditions	0 = real data case (default) 1 = GDEM seasonal climatology 2 = constant density
clm_scl	real	climate decorrelation time scale (hours) for ice, sst, ssh, mvoi, and swh.	default = 480, 720, 720, 720, 96
cold_start	logical	force cold start on nest 1 (true)	default = false
debug	logical	generate diagnostics (true) for 1) pooling of profile moorings and profile rejections (fort.32), 2) profile inflexion point and standard level data and vertical extension results using background fields (fort.33), 3) geopotential profile observations (fort.34), 4) observation and prediction errors (fort.35), 5) listing of MVOI observations (fort.36), 6) synthetics (direct and MODAS) for MODAS including rejections and editing results (fort.31)	default = false (for all)
dbv_opt	char	source bathymetry database	DOD = DBDBV from NAVOCEANO (default) SAS = Smith and Sandwell all = all sources (DBDBV plus Smith and Sandwell)
dbv_res	real	minimum resolution of bathymetry to retrieve from variable resolution database (minutes)	default = 5
del_ssh	real	minimum change in SSHA to force generation of a synthetic profile	default = 0.02
del_sst	real	minimum change in SST to force sampling of analyzed SST field	default = 0.2
den_ds	real	change in density definition for MLD	default = 0.15
deny	integer	data types to deny analysis (see "coda_types.h" for type codes)	default = 0
direct	logical	(true) perform direct assimilation SSHA	default = false
dv_dz	real	vertical gradient length scale (units are dependent upon vc_mdl selection)	default = 0.3
ebkg	real	background error tuning factors for ice, sst, ssh, temperature, salinity, geopotential, velocity, and swh.	default = 1 (for all)
eobs	real	obs error tuning factors (see "coda_types.h" for type codes)	default = 1 (for all)

COAMPS Ocean Analysis *oanl* Namelist (continued)

Name	Type	Description	Values
emdl	char	background error option for ice, sst, ssh, temperature, salinity, geopotential, velocity, and swh.	'simple' = homogeneous errors (default for all) 'complx' = non-homogeneous errors
hc_mdl	char	horizontal correlation model options for ice, sst, ssh, mvoi, and swh.	'rsby' = non-homogeneous scales (default for all) 'homo' = homogeneous scales 'locl' = user defined scales
linck	logical	perform innovation error check (true) for ice, sst, ssh , temperature, salinity, geopotential, velocity, direct method synthetics, and swh.	default = false (for all)
linit	logical	(true) perform initialization procedures on cold start fields	default = false
lncom	logical	(true) perform NCOM setup procedures	default = false
locn3d	logical	(true) do 3D analysis on this grid nest	default = false (for all grid nests)
mask_opt	char	grid mask option ("2D" or "3D") (used only when locn3d is true)	default = "2D"
mds_age	real	minimum obs age (hrs) to force synthetic	default = 144
mds_edit	logical	(true) edit MODAS synthetics	default = true
mds_grd	logical	(true) generate MODAS synthetic profile initial conditions on a cold start	default = false
mds_mld	logical	(true) apply modas mld model	default = true
mds_xtnd	logical	(true) extend MODAS synthetics with Levitus climatology	default = true
pool	logical	(true) pool satellite systems for MSP F11, F13, F14, F15; NOAA 14, 15, 16, 17 GAC SSTs; TOPEX, ERS, GFO, JASON, ENVISAT; GOES 8, 10 SSTs; NOAA 16,17 LAC SSTs; Altimeter / Buoy SWH	default = false (for all)
prf_opt	char	profile processing option	obsz = assimilation of profiles on observed levels anlz = assimilation of profiles after interpolation to analysis levels (default)
prf_slct	real	profile selection criteria options for 1) acceptable level probability gross error 2) minimum number of sampling levels 3) minimum ratio of last sampling depth and bottom depth 4) minimum sampling depth (if profile has not sampled water column) 5) maximum acceptable distance between adjacent levels 6) maximum acceptable temperature difference between adjacent levels	default = 0.99, 5, 0.5, 300, 300, 5

COAMPS Ocean Analysis *oanl* Namelist (continued)

Name	Type	Description	Values
prf_time	char	profile time sampling option	obst = select profiles based on time profile was observed (default) rcpt = select profiles based on time profile was received at center
prf_xtnd	logical	(true) extend inflexion point profiles to bottom using first guess fields	default = true
qc_err	real	max acceptable probability gross error for 1) DMSP sea ice 2) AVHRR satellite sst 3) GOES satellite sst 4) in situ sst 5) profile temperature (integrated over levels) 6) profile salinity (integrated over levels) 7) altimeter SSHA 8) altimeter/buoy SWH	default = 0.99, 0.99, 0.99, 0.99, 0.99, 0.99, 1, 0.99
rscl	real	rossby radius scaling factor for ice, sst, ssh, multivariate, and swh.	default = 3 (for all)
sal_adj	logical	(true) adjust SSS derived from SST obs for density inversions	default = true
sal_std	real	max number std dev to scale modas salinity from climatology	default = 3
spval	real	missing value (must be < -99)	default = -999
ssh_time	char	altimeter ssh processing option	obst = select obs based on time SSHA was observed (default) rcpt = select obs based on time SSHA was received at center cycl = select obs based on full Topex repeat cycle (10 days)
ssh_std	real	max number std dev to scale altimeter ssh from climatology	default = 1
st_chn	logical	(true) generate "thermistor chain" observations to the base of the mixed layer from SST (see also st_grid)	default = false (for all grid nests)
st_grd	logical	(true) generate SST observations for 3D MVOI from analyzed SST grid	default = false (for all grid nests)
st_ntrvl	integer	SST "thermistor chain" vertical sampling interval	default = 1
swh	logical	(true) perform SWH analysis	default = false
swh_time	char	swh observation processing option	obst = select obs based on time swh observed (default) rcpt = select obs based on time swh received at center
tmp_std	real	max number std dev to scale modas temperature from climatology	default = 4
tol_fctr	real	innovation error check tolerance factor for ice, sst, ssh, temperature, salinity, geopotential, velocity, and swh.	default = 4 (for all)

COAMPS Ocean Analysis *oanl* Namelist (continued)

Name	Type	Description	Values
upd_cyc	integer	analysis update cycle (hours)	default = 24
vc_mdl	char	vertical correlation model options	'mixl' = mixed layer (default) 'dens' = density stratification 'temp' = temperature stratification 'mono' = monotonic 'cons' = constant 'none' = no vertical correlation
vol_scl	real	minimum number of correlation length scales in an analysis volume for ice, sst, ssh, mvoi, and swh.	default = 4 (for all)
z_lvl	real	analysis vertical grid (meters).if lncom is true, then the vertical grid is computed from ncom model parameters specified in "omnl.h" and z_lvl is ignored	default = 0, 5, 10, 15, 20, 30, 50, 75, 100, 125, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900,1000, 1100, 1200, 1300, 1400, 1500, 1750, 2000, 2500, 3000, 70*0

COAMPS *gridnl* Namelist

Name	Type	Description	Default Values
alnnt	real	standard longitude of grid	240
delx	real	grid spacing in x-direction	81000
dely	real	grid spacing in y-direction	81000
ii	integer	coarser mesh x grid point	1 (for all grids)
iref	integer	i-coordinate of reference point	1 (for all grids)
jj	integer	coarser mesh y grid point	1 (for all grids)
jref	integer	j-coordinate of reference point	1 (for all grids)
kka	integer	number atm vertical levels (layers)	30
kko	integer	number ocn vertical levels (layers)	1
lnmove	logical	(true) moving nest indicator	false (for all grids)
m	integer	number x grid positions	61
n	integer	number y grid positions	61
nnest	integer	number nested grids	1
npgrid	integer	parent grid number	1,1,2,3,4,5,6
nproj	integer	grid projection number	2
phnt1	real	1st standard latitude of grid	60
phnt2	real	2nd standard latitude of grid	30
rlat	real	grid reference latitude	42.5
rlon	real	grid reference longitude	16.5

***coamnl* Variables required by the COAMPS Radiation Module**

Name	Type	Description	Default Value
lrad	Logical	Turn on the radiation calculation. = t, turn on both longwave and shortwave calculation = f, turn off the radiation calculation	t
dtrad	Real	Time interval in seconds to update radiative heating rate by calling "radiat".	3600
njump	Integer	Grid interval in longwave calculation.	1
ldgrad	Logical	Print diagnostic output of longwave and shortwave fluxes at a selected point (*idgrad, jdgrad*). = t, print diagnostics = f, do not print	false
idgrad	Integer	I-coordinate of the selected point for *ldgrad*.	2
jdgrad	Integer	J-coordinate of the selected point for *ldgrad*.	2

***aeromnl.h* Aerosol-Tracer Namelist**

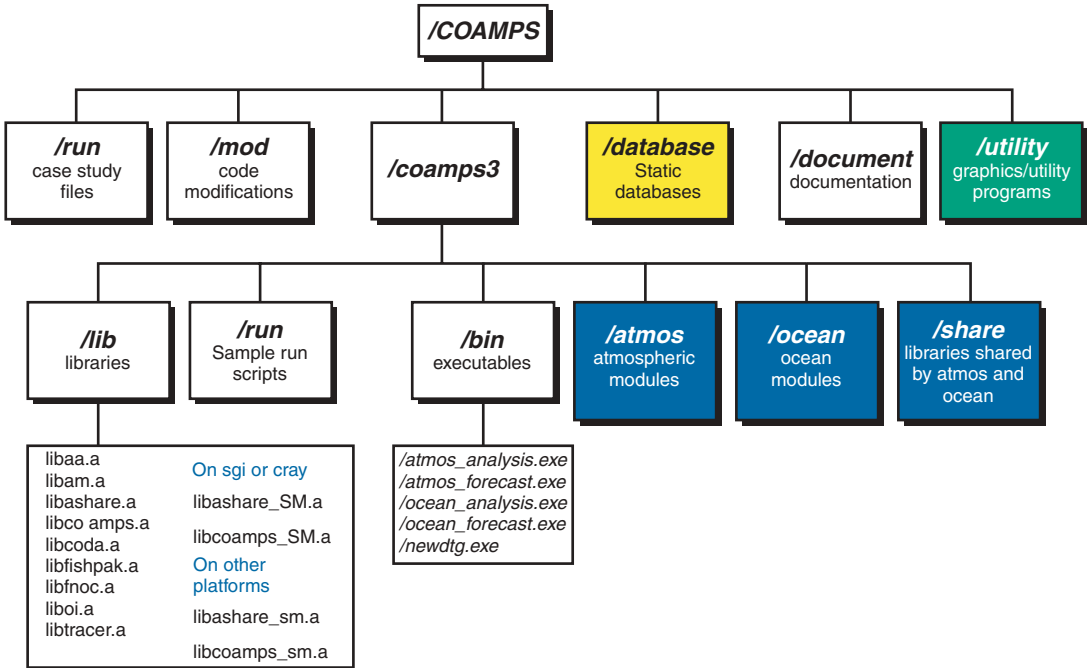
Name	Type	Description	Default Value
kcctyp	integer	Type of output time. = 1, output at specified time = 2, output at a frequency.	2
kccfrq (3)	integer array	Frequency or time interval of output (hr, min, sec).	N/A
kcctime (3,kcctot)	integer array	Specify output time (hr, min, sec) with *kcctot* number of output.	kcctot = 500
nadvtyp	integer	Choice of advection schemes used for transport. = 1, Bott's 5th-order flux-form advection = 2, Bott's 7th-order flux-form advection	1
emssflg	integer	Type of source functions. = 0, no mass emission or production = 1, general point- or area-source tracer emissions = 2, dust mobilization only = 3, other type source functions in future	1
nintcc	integer	Type of concentration initialization. = 0, zero initial concentration field = 1, user provides initial concentration field	0
irrgsrc	integer	Type of emission setup, used when *emssflg* =1. = 0, regular setup by given *nsrc*, *cntlat*, *cntlon*, *cnthgt*, etc. = 1, user provides specific or real case emission-related data	0
nsrc	integer	Number of source emission sites or locations, used with *emssflg* = 1 and *irrgsrc* = 0.	nsrc = 1
cntlat (nsrc)	real array	Latitudes of emission site centers, used with *emssflg* = 1 and *irrgsrc* = 0.	N/A
cntlon (nsrc)	real array	Longitudes of emission site centers, used with *emssflg* = 1 and *irrgsrc* = 0.	N/A
cnthgt (nsrc)	real array	Heights (m) of emission site centers above the ground, used with *emssflg* = 1 and *irrgsrc* = 0.	cnthgt = 10
ntems	integer	Number of emission-strength changes in time for instantaneous and continuous releases, used with *emssflg* = 1 and *irrgsrc* = 0 or 1.	1, continuous emission with constant strength

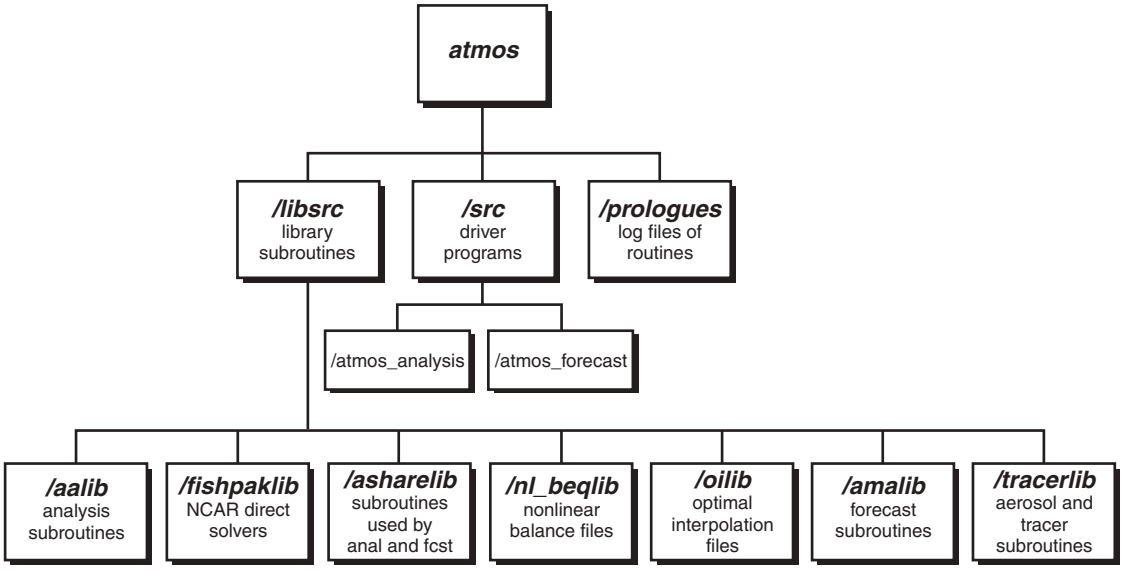
***aeromnl.h* Aerosol-Tracer Namelist (continued)**

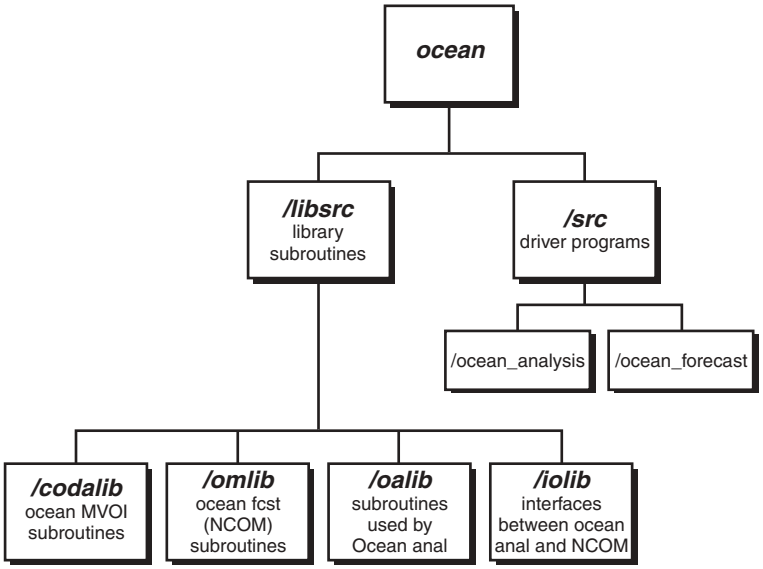
Name	Type	Description	Default Value
emstime (3,ntems)	integer array	Time list (hr, min, sec) at which emission strength changes for *ntems* numbers. Used with *emssflg* = 1 and *irrgsrc* = 0 or 1.	N/A
emstrng (ntems,nsrc)	real array	Emission strength or flux (kg/sec) at each time for each emission site. Used with *emssflg* = 1 and *irrgsrc* = 0 or 1.	N/A
ngrdems	integer	Type of emission site in spatial size. Used with *emssflg* = 1 and *irrgsrc* = 0. = 1, emission from single grid volume = 2, emission from a multiple grid volume, i.e. a 3-d round- or oval-shape cloud	1
hhrad (nsrc)	real array	Horizontal radius (m) of cloud-shape volume for each site, used with *ngrdems* =2.	N/A
zzrad (nsrc)	real array	Vertical radius (m) of cloud-shape volume for each site used with *ngrdems* = 2.	N/A
ccycle	logical	Type of model start, cold or warm.	true
aerotme0 (3)	integer array	Beginning time (hr, min, sec) of the aerosol-tracer integration. (0 ≤ *aerotme0* ≤ COAMPS forecast time)	0,0,0
lwaycc	integer	Type of grid nest interaction. = 0, no grid nest interaction = 1, feedback from high-resolution grid nest to low-resolution grid = 2, feedback from low-resolution grid nest to high-resolution grid. = 3 two-way, including 1 and 2.	0
lmbudg	logical	Flag of detailed mass budget calculation used for program tests. = true, yes = false, no	false
ldrydep	logical	Flag of aerosol dry deposition calculation. = true, yes = false, no	false
lwetdep	logical	Flag of aerosol wet deposition calculation. = true, yes = false, no	false
lsedimn	logical	Flag of aerosol sedimentation calculation. = true, yes = false, no	false
ldosage	logical	Flag of mass dosage calculation and output (mg*min/m**3). = true, yes = false, no	false
loptic	logical	Flag of aerosol optical thickness calculation and output. = true, yes = false, no	false
lmsload	logical	Flag of mass load (vertical integral) calculation and output (mg/m**2). = true, yes = false, no	false

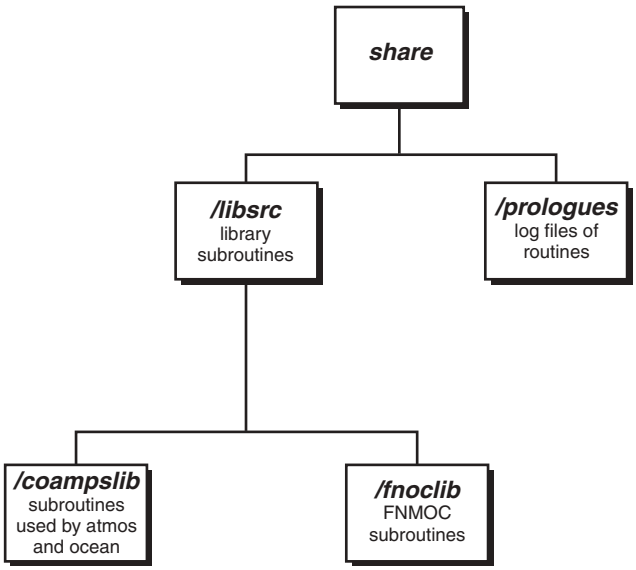
Appendix C

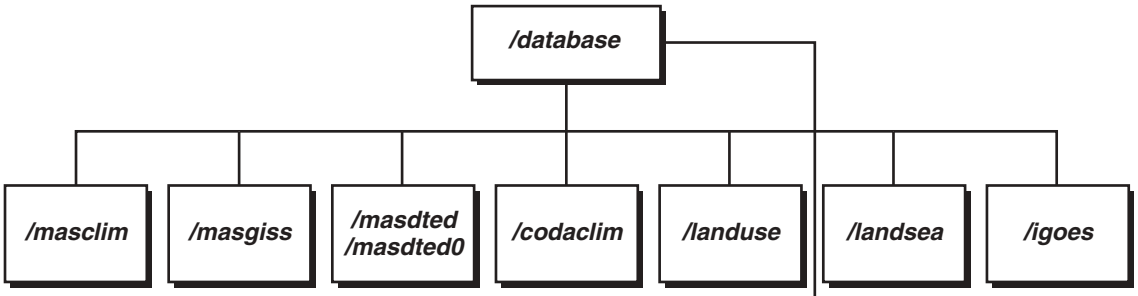
COAMPS Directory Tree





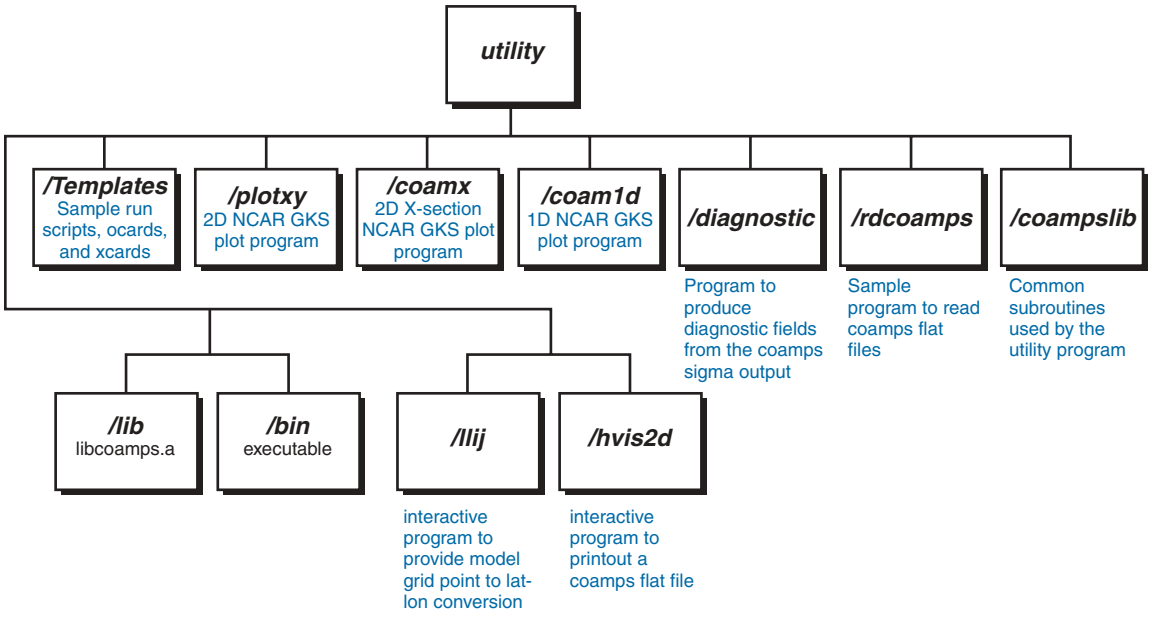

ocean
/libsrc
library
subroutines
/src
driver programs
/ocean_analysis
/ocean_forecast
/codalib
ocean MVOI
subroutines
/omlib
ocean fcst
(NCOM)
subroutines
/oalib
subroutines
used by
Ocean anal
/iolib
interfaces
between ocean
anal and NCOM


share
/libsrc
library
subroutines
/prologues
log files of
routines
/coampslib
subroutines
used by atmos
and ocean
/fnoclib
FNMOC
subroutines



1. masclim: global climatology surface characteristics data (resolution vary depends on what fields)
2. masdted or masdted0: 1 km global terrain data.
3. masgiss: one degree NASA GADDAR high resolution albedo and z0 data.
4. codaclim: CODA sst and ice climatology files.
5. goes: GOES 8 and GOES 9 satellite channel data to compute brightness temperature.
6. landsea: 400 meter resolution land-sea table.
7. landuse: 1 km global landuse data.

1. ocnqc: observation data used by CODA for bench case (1999083100-1999090700 and 1999120800-1999121500).
2. nogaps: nogaps data for bench use.
3. adp: adp data for bench case.



Appendix D

COAMPS Sample Run Script

```
#!/bin/sh
#
#####################################################################
#####################################################################
#      Script to run COAMPS
#      run.bench $1 $2
#      $1 is the current date-time-group
#      $2 = 2 : run forecast model only
#####################################################################
#####################################################################

if [ "$#" -lt 2 -o "$#" -eq 3 ]; then
   clear
   echo
"Usage: run.coamps def1 def2 def3 def4"
   echo "  where "

   echo "       def1: beginning date-time-group yyyymmddhh"
   echo " "

   echo "       def2:   0, 1, 2, or 3 "
   echo "            0- no action"
   echo "            1- execute COAMPS analysis"
   echo "            2- execute COAMPS forecast"
   echo "            3- both 1 and 2"
   echo " "

   echo "       def3: end update cycle date-time-group yyyymmddhh (optional)"
   echo " "

   echo "       def4: update cycle interval hh (optional)"
   echo " "
   exit

elif [ "$#" -eq 2 ]; then
    d3=$1
    d4=12
else
    d3=$3
    d4=$4
fi
#
if [ -s co.$1.sh ]
  then
  rm -f co.$1.sh
fi
#



#####################################################################
# generate coamps run script
#####################################################################
#
#
# set number of processors for the forecast model
# name:         the name of this run script
#
name=`basename $0`
NPROC=5
NX=2,2,2,2,2,2,2
NY=2,2,2,2,2,2,2
Interactive=no

#
#####################################################################
# set up interactive commands
#####################################################################
#
if [ $Interactive = "yes" ]; then
  cat > co.$1.sh << EOF
!/bin/ksh
EOF
fi
#
#####################################################################
# set up batch commands
#####################################################################
#

#
#####################################################################
#   SGI DALEY
#####################################################################
#
if [ `uname -n` = "daley" ]; then

batch="qsub"

cat > co.$1.sh << EOF
#!/bin/ksh
#PBS -j oe
#PBS -l ncpus=${NPROC}
#PBS -o $name.log.$1
#PBS -q short

set -x

PATH=\$PATH:/usr/contrib/nrl/bin
export PATH
EOF

#
#####################################################################
#   SGI AMS1 or AMS2
#####################################################################
#
elif   [ `uname -n` = "ams1" -o `uname -n` = "ams2" ];then

batch="bsub"

cat >> co.$1.sh << EOF
#!/bin/ksh
#BSUB -J $name.$1
#BSUB -c 10:30
#BSUB -n ${NPROC}
#BSUB -o $name.$1.${NPROC}p.out
#BSUB -q medium
EOF

#
#####################################################################
#   NAVO DEC ALPHA (opal)
#####################################################################
#
elif [ `uname -n` = "opal0" -o `uname -n` = "opal1" ]; then

batch="bsub"

cat >> co.$1.sh << EOF
#BSUB -J $name.$1                  #job name
#BSUB -n ${NPROC}                    #total no of processors
#BSUB -W 00:50                     #request time
#BSUB -o co.$1.log.${NPROC}p.%J   # stdout and stderr output file name
#BSUB -q primary                   #queue name
#BSUB

set -x
hostname
EOF

#
#####################################################################
#   NAVO IBM SP (habu)
#####################################################################
#
elif [ `uname -n` = "habu"]; then

batch="llsubmit"

NODE=`expr $NPROC \/ 4`
  if [ $NODE -eq 0 ]; then
```

```
   NODE=1
 fi

cat > coamps.log.$1 << EOF
#@ shell=/bin/ksh
#@ class=batch
#@ wall_clock_limit=4:00:00
#@ job_type=parallel
#@ node_usage=not_shared
#@ output=$name.$1.log.${NPROC}p
#@ error=$name.$1.log.${NPROC}p
#@ job_name=co.$1.sh
#@ node=$NODE
#@ tasks_per_node=4
#@ network.MPI=switch,shared,US
#@ account_no=NRLMR024
#@ notification=never
#@ environment=COPY_ALL
#@ queue
#
set -x

#
#########################################################################
#
ARL SGI
#########################################################################
#
elif [ ‘uname -n‘ = “adele”  -o ‘uname -n‘ = “adele1” \
   -o ‘uname -n‘ = “zornig” -o ‘uname -n‘ = “herman” ]; then

batch=”qsub”

cat >> co.$1.sh << EOF
#$ -S /bin/ksh
#$ -pe pe_96hr ${NPROC}
#$ -eo
#$ -l sgi
#$ -cwd
EOF

#
#########################################################################
#    ARL IBM
#########################################################################
#
elif [ ‘uname -n‘ = “brainerd1” -o ‘uname -n‘ = “brainerd2” ];then
cat >> co.$1.sh << EOF

batch=”qsub”

#$ -S /bin/ksh
#$ -P NRLMR03795024
#$ -pe mpi_4hr_ibm $NPROC
#$ -eo
#$ -l ibm
#$ -cwd

set -x

EOF

fi

cat >> co.$1.sh << EOF

nproc=$NPROC
nx=$NX
ny=$NY

EOF
cat >> co.$1.sh << ‘EOF’

  banner ‘uname -n‘
#
# databaseDir:   directory path where COAMPS database is located
# nogapsDir:     directory path of NOGAPS flat files.
# adpDir:        directory path of ADP files
# osname:        coamp make target name (sgi, ibm, or alpha)
#
   scrDir=/rdisk/1/rdata1/$LOGNAME
   database=/coamps/database
   nogapsDir=/coamps/database/nogaps
   adpDir=/coamps/database/adp
   ocnDir=/coamps/database/ocnqc

#
# number of OMP threads for analysis
#
   OMP_NUM_THREADS=$nproc
   OMP_SCHEDULE=GUIDED
   MP_STACK_SIZE=800000000
   export OMP_NUM_THREADS
   export OMP_SCHEDULE
   export MP_STACK_SIZE
#########################################################################
#   define variables needed for run
#########################################################################
#
# expname:   user’s model experiment name
# modDir:    directory path under which coamps source code is compiled
# runDir:    directory path where the run time print out is saved
# binDir:    directory path where the coamps executable files are located
# saveDir:   directory path where the model output is saved
# Inocards:  ocards file name
# Inxcards:  xcards file name
# recompile: yes - regenerate the executable
# ddtg:      start time of the forecast yyyymmddhh

  expname=bench_summer
  saveDir=$scrDir/coamps/$expname
  dataDir=$scrDir/coamps/data/$expname
  modDir=$HOME/coamps/mod/$expname
#########################################################################
#########################################################################
#   set program names
#########################################################################

  binDir=$HOME/coamps/coamps3/bin
  newdtg=$binDir/newdtg.exe
  analysis=$binDir/atmos_analysis.exe
  forecast=$binDir/atmos_forecast.exe
  if [ ‘uname -n‘ = “daley” ]; then
    runDir=$PBS_O_WORKDIR
  else
    runDir=‘pwd‘
  fi

  recompile=no
  get_nogaps=yes
  get_adp=yes
  get_ocn=yes
  get_coamps=yes
  clean_data=yes
  save_data=yes
  Ocards=ocards.$expname
  Xcards=xcards.$expname
#
#  load bach queing utility - grd (arl)
#
  if [ -e /usr/grd/default/common/settings.sh ]; then
     . /usr/grd/default/common/settings.sh
  fi
#
#  setup mpi environment and execution commands
#
  if [ ‘uname -s‘ = “IRIX64” ]; then

#
#  load sgi modules
#
  if [ -e /opt/modules/modules/init/ksh ]; then
    . /opt/modules/modules/init/ksh
    module load MIPSpro mpt scsl
    module list
  fi

    forecast=”mpirun -np $nproc $forecast”
#    forecast=”totalview mpirun -a -np $nproc $forecast”
    echo “runnin on sgi”
    osname=sgi
    unlimit stacksize
    limit coredumpsize      0
    limit datasize          unlimit
    limit memoryuse         unlimit

    export MP_SLAVE_STACKSIZE=$(( 1024 * 1024 * 1024 ))

    MPI_TYPE_MAX=81920
    export MPI_TYPE_MAX
    SMA_GLOBAL_ALLOC=1
    export SMA_GLOBAL_ALLOC

    SMA_GLOBAL_HEAP_SIZE=512753664
    export SMA_GLOBAL_HEAP_SIZE

    printenv | grep -e _DSM_ -e MPI_ -e MP_ -e OMP_

  elif [ ‘uname -s‘ = “AIX” ];then

    osname=ibm
    newdtg=”grd_poe $newdtg”
    forecast=”grd_poe $forecast”
    echo “runnin on ibm”
```

```
   XLFRTEOPTS="namelist=OLD"
   XLSMPOPTS="SCHEDULE = GUIDED : parthds = 1 : stack = 900000000"
   MP_LABELIO=no

   export XLFRTEOPTS
   export XLSMPOPTS
   export MP_LABELIO

  elif [ `uname -s` = "OSF1" ];then

   osname=alpha
   if [ `uname -n` = "opal0" -o `uname -n` = "opal1" ]; then
     forecast="prun -n $nproc $forecast"
   else
     forecast="dmpirun -np $nproc $forecast"
#    forecast="totalview dmpirun -a -np $nproc $forecast"
   fi
   echo "runnin on alpha"

  elif [ `uname -s` = "Linux" ];then
   osname=linux
# LAM using the LAM daemon for communications
#      forecast="mpirun -x CRDATE -lamd -np $nproc $forecast"
# LAM without daemon (faster speed)
      forecast="/usr/local/lam/bin/mpirun -x CRDATE -c2c -np $nproc $forecast"
# MPICH
#      forecast="mpirun -np $nproc $forecast"

  fi
EOF

cat >> co.$1.sh << EOF
def1=$1
def2=$2
def3=$d3
def4=$d4
bdate=$1
edate=$d3
runscript=$name
EOF

cat >> co.$1.sh << 'EOF'
echo " "
echo "user defined begin date-time-group: $def1"
echo "user defined end   date-time-group: $def3"
echo " "
echo "======================================================="
#
# run through update cycle
#

  ddtg=$def1
  ddtgm1=`echo $ddtg -$def4 | $newdtg`
  ddtgp1=`echo $ddtg $def4 | $newdtg`
  echo "running date-time-group $ddtg"
  year=`echo $ddtg | cut -c1-4`
  month=`echo $ddtg | cut -c5-6`
  day=`echo $ddtg | cut -c7-8`
  ocnddtg=${year}${month}${day}00
  ocnddtgm1=`echo $ocnddtg -96 | $newdtg`
  ocnddtgp1=`echo $ocnddtg 24 | $newdtg`
  ocnddtg=$ocnddtgm1

#########################################################################
#########################################################################
#   go to directory in which to run job
#########################################################################
#########################################################################

  if [ ! -d $runDir ]
    then
    mkdir -p $runDir
  fi

  cd $runDir

#########################################################################
#########################################################################
#   clean up unwanted files at beginning
#########################################################################
#########################################################################

  if [ -s anal.$ddtg.$expname ]; then
     rm -f anal.$ddtg.$expname
  fi

  if [ -s fcst.$ddtg.$expname ]; then
     rm -f fcst.$ddtg.$expname
  fi

  if [ -s oanl.$ddtg ]; then
     rm -f oanl.$ddtg
  fi

#########################################################################
#   create the namelist inputs for the analysis and forecast
#########################################################################

# Model domain grid information
#
# kka:     no. of vertical height levels in the model
# m:       1 d array, number of grid points in the x direction for each grids
# n:       1 d array, number of grid points in the y direction for each grids
# nnest:   maximum number of grids in the forecast
# npgrid:  1 D array (one for each grid) that defines the parent grid
#          id (coarse domain is always 1).
# nproj:   map projection identifier
#          1: Mercator; 2:lambert conformal; 3:polar stereographic;
#          4: Cartesian coordinates; 5: spherical
# alnnt:   longitude aligned north-south; used for rotating a lambert conformal
#          or polar stereographic projection
# alpha:   averaging coefficient for semi-implicit time
# phnt1:   1st latitude where map projection intersects the earth
# phnt2:   1st latitude where map projection intersects the earth
# rlat:    standard latitude used to set up coarse domain
# rlon:    standard longitude used to set up coarse domain
# iref:    coarse domain x grid point which correspond to rlat1
# jref:    coarse domain y grid point which correspond to rlong1
#
  cat << eof > gridnl.$ddtg
 &gridnl
 kka   =  30,
 m     =  72,67,91,
 n     =  45,67,91,
 nnest =  2,
 npgrid=  1,1,2,
 nproj =    2,
 alnnt =    286.0,
 phnt1 =    60.0,
 phnt2 =    15.0,
 rlat  =    38.0,
 rlon  =    286.0,
 iref  =    50,
 jref  =    25,
 ii    =    1,35,16,
 jj    =    1,10,16,
 delx  =    81000.0,
 dely  =    81000.0,
 &end
eof

# nbdya: real data lateral boundary condition identifier for nests
#        6 - Perkey-Krietzberg
#        7 - Davies
# nbdypt:number of points included in lateral boundary computation
# ndxnam: number of processors used in the x direction
# ndynam: number of processors used in the y direction
# npr0nam: 0 - parallel mpi I/O (mpi1 or mpi2)
#          1 - single mpi I/O (which uses a dedicated I/O processor for
#              MPI-1 only). When use this option, increase the number
#              of processors in your script by 1
#
  cat << eof > atmosnl.$ddtg
 &atmosnl
 nbdya =  7,
 nbdypt=  7,
 ndxnam    =  $nx,
 ndynam    =  $ny,
 nbnam     =  2,
 npr0nam   =  1,
 loopvecnam =  0,
 lmbc   =  21,
 idelay  =  0, 0, 0,
            0, 0, 0,
            0, 0, 0,
 iaero  =  0,
 lgrdall=  f,
 nestcc =  2,
 &end
eof

# Input/output files information
#
# dsclim: directory path of climatology surface characteristics data.
# dsetng: isis directory path of NOGAPS data input/output.
# dsngff: directory path of NOGAPS flat files.
# dsnrff: directory path of COAMPS flat files.
# dsgiss: directory path of GADDAR high resolution albedo and z0 data
# dsdted: directory path of 1 km terrain data
#         masdted:  YNIMA DTED1 1km terrain database
#         masdted0: YNIMA DTED0 1km terrain database
# dsland: directory path of 400 meter land-sea tables
# dslanu: directory path of 1 km landuse data
# dsoudat: directory path of unclassified ocean observations
# dsordat: directory path of restricted ocean observations
# dsocdat: directory path of confidential ocean observations
# dsosdat: directory path of secret ocean observations
```

```
# dsoclim: directory path of ocean databases
# dsorff:  directory path of ocean flat files
# dsomdas: directory path of MODAS database files
# npfil:  name of file containing 2D horizontal level plotting information
# xsfil:  name of file containing 2D cross-sectional plotting information

 cat << eof > dsetnl.$ddtg
&dsetnl
 dsclim  = '$database/masclim/',
 dsetng  = 'fcst_ops',
 dsngff  = '$scrDir/f$ddtg/',
 dsnrff  = '$dataDir/',
 dsgiss  = '$database/masgiss/',
 dsdted  = '$database/masdted/',
 dsland  = '$database/landsea/',
 dslanu  = '$database/landuse/',
 dsoudat = '$scrDir/ocnqc',
 dsoclim = '$database/codaclim',
 dsomdas = '$database/modas',
 &end
eof

 cat << eof > oanl.$ddtg
&oanl
 emdl    = 8*'complx',
 hc_mdl  = 5*'rsby',
 locn3d  = .false.,
 rscl    = 3., 4., 4., 4.,
 upd_cyc = 12,
 &end
eof
# Model configuration
#
# lcoda    t: use CODA sst and sea ice analysis
#          f: use NOGAPS
# icase:   0: real data model simulation
#            idealized model simulation
#          3: hurricane
#          8: Knupp convection
# ibdya, jbdya: lateral boundary conditions for x and y direction
#          for idealized
#           1: fixed; 2: extrapolated; 3:
periodic; 4: radiation;
#           5: radiation and extrapolation
#          for real data
#           6: Perkey-Krietzberg
#           7: Davis
# kgetbc:  frequency for reading coarse domain boundary tendencies
# flat:    latitude; used for Cartesian grids to compute the coriolis
# delta:   model coarse domain (large) time step (second)
# ktaust:  starting time of forecast (hr, min,sec)
# ktauf:   ending time of forecast (hr, min,sec)
# ii, jj:  1D array; mother domain i and j points that defines the nested domains
# delx:    coarse domain grid spacing (meter) in x direction
#          for spherical map projection, the unit for delx is degree
# dely:    coarse domain grid spacing (meter) in y direction
#          for spherical map projection, the unit for dely is degree
# dsigma:  1D array (dimension of kka); vertical grid spacing (meter)

 cat << eof > coamnl.$ddtg
&coamnl
 npfil  = '$Ocards',
 xsfil  = '$Xcards',
 ncast_start = 0 ,
 ncast_end   = 24,
 ncast_cycle = 12,
 lcoda  =     t,
 icase  =     0,
 ibdya  =     7,
 jbdya  =     7,
 kgetbc =     6,
 alpha  =     1.0,
 flat   =    45.0,
 delta  =   240.0,
 ktaust =     0,  0,  0,
 ktauf  =    24,  0,  0,
             24,  0,  0,
             24, 0,  0,
 dsigma =    7500.0, 5800.0, 4200.0, 2500.0, 1000.0,
             1000.0, 750.0,  750.0,  750.0,  750.0,
             750.0,  750.0,  1000.0, 1000.0, 1000.0,
             1000.0, 800.0,  800.0,  800.0,  600.0,
             400.0,  300.0,  200.0,  140.0,  90.0,
             60.0,   40.0,   30.0,   20.0,   20.0,
eof

#
# moving nest option
#
# Moving nests can be turned on in coamps through the gridnl namelist
# logical parameter lnmove=t (function of nest)
# imovtyp= 1 - must specify the number of grid points
#            (nmovei,nmovej) and times (itmove) to move each nest
#       = 2 - specify locations and times where and when
#             you want a specified nest to be centered
# itmove =    3000*-1, this says no moves for nest 1 (1000*3 values of -1)
#                always true
#         2, 0, 0,  this says first move is at 2H, 0M, 0S
#         6,24, 0,  next move is at 6H, 24M
#         8, 0, 0,  last move is at 8H, 0M
# nmovei = 1000*0,   nest 1 doesn't move in x-dir (ever)
#           2,-1,1,  nest 2 moves 2, -1, and 1 grid points for the
#                 above three times positive is to the east
# nmovej  = 1000*0,       nest 1 doesnt' move in y-dir (ever)
#           1, 2,-1,     nest 2 moves 1, 2, and -1 grid points for the above
# itmove  = 3000*-1,
  no moves for nest 1
#           22, 0, 0,    at 22H, 0M, nest 2 will be centered at
#                  (xlatmov,xlonmov)
# xlatmov = 1000*-999.9,  no moves for nest 1
#           45.0,        nest 2 center latitude will be 45.0
# xlonmov = 1000*-999.9,  no moves for nest 1
#           20.0,        nest 2 center longitude will be 20.0

 cat << eof >> coamnl.$ddtg
 imovtyp=     1,1,1,
 itmove =     3000*-1,
              3000*-1,
              2, 0, 0,
              6,24, 0,
              8, 0, 0,
 nmovei  = 1000*0,
           1000*0,
            2,-1, 1,
 nmovej  = 1000*0,
           1000*0,
           1, 2,-1,
 xlatmov = 1000*-999.9,
           45.0,
 xlonmov = 1000*-999.9,
           20.0,
eof
# for idealized sounding input (sfc to top of the atmosphere)
#
# psnd:  ipsnd=1, pressure in mb
#        ipsnd=2, height in meter
#
# tsnd:  itsnd=1, temperature in Kelvin
#        itsnd=2, temperature in C
#        itsnd=3, temperature in theta
#
# qsnd:  iqsnd=1, dewpoint in C
#        iqsnd=2, dewpoint in K
#        iqsnd=3, moisture in g/kg
#        iqsnd=4, relative humidity (%)
#
# usnd:  iusvnd=1, u velocity (m)
#        iuvsnd=2, direction of the wind (degree)
#        iuvsnd=3, linear wind profile based on usnd(1) and usnd(top)
#
# vsnd:  iuvsnd=1, v velocity (m)
#        iuvsnd=2, speed of the wind (m)
#        iuvsnd=3, linear wind profile based on vsnd(1) and vsnd(top)
#
# psfc0: sea level pressure; used to compute surface pressure
# tamp:  amplitude of idealized temperature perturbation (icase=8)
# laper: T - aperiodicity at boundaries
# lasym: T - asymmetry in field
# lvpert:T - perturb v fields (icase=13 or 14)
# jcm2fg:0 - use reference sounding to initialize nest
#        1 - interpolate coarse domain initial condition to the nest
# umean: mean u wind speed (m/s) to be subtracted from the sounding
# vmean: mean v wind speed (m/s) to be subtracted from the sounding

 cat << eof >> coamnl.$ddtg
 ipsnd =    1,
 iqsnd =    3,
 itsnd =    1,
 iuvsnd =   2,
 psnd=1013.08,  1012.52, 1008.18, 999.77, 987.58, 971.88, 952.92, 930.96,
      906.23,  878.97,  849.40, 817.72, 784.14, 748.85, 712.02, 673.83,
      634.44,  594.00,  552.64, 510.50, 467.70, 424.35, 380.54, 336.37,
      291.92,  247.26,  202.44, 157.52, 112.54,  67.53,  22.51, 1.,
 tsnd= 291.04,  291.,    290.77, 290.29, 289.6,  288.7,  287.6,  286.27,
      284.76,  283.06,  281.15, 279.05, 276.74, 274.21, 271.47, 268.48,
      265.25,  261.74,  257.93, 253.8,  249.28, 244.32, 238.86, 232.77,
      225.93,  218.1,   208.94, 197.88, 183.8,  163.99, 127.61, 127.61,
 usnd =270.,    270.,    270.,   270.,   270.,   270.,   270.,   270.,
      270.,    270.,    270.,   270.,   270.,   270.,   270.,   270.,
      270.,    270.,    270.,   270.,   270.,   270.,   270.,   270.,
      270.,    270.,    270.,   270.,   270.,   270.,   270.,   270.,
 vsnd =  0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
         0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
         0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
         0.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
 qsnd =  3.,     3.,     2.,     1.,     1.,     1.,     .1,     .1,
```

```
       0.,    0.,    0.,    0.,    0.,    0.,   0.0,   0.0,
       0.,    0.,    0.,    0.,    0.,    0.,   0.0,   0.0,
       0.,    0.,    0.,    0.,    0.,    0.,   0.0,   0.0,
 psfc0  =  101325.0,
 tamp   =      0.0,
 laper  = f,
 lasym  = f,
 lvpert =    f,
 jcm2fg = 1,
 umean  = 0.0,
 vmean  = 0.0,
eof
#
# numeric options
#
# iadvct:  1: 4th order; 2: 2nd order advection
#          3: 2nd order with 2nd oder upstream for theta
#          4: spectrl
#          5: 2nd order flux
#          6: mixed; 2nd order advective and flux
# lspong:  T - damp (u,v,w,theta) to smoothed values in the top nrdamp levels
#          F - do not do above
# rdtime:  time scale to damping u,v,w,theta in sponge layer
# nrdamp:  number of upper model levels that are in the sponge layer
# lralee:  raleigh damping layer
# robert:  coefficient for coupling the three time levels; used to filter high
#          frequency oscillations
# lddamp:  T - divergence damping on
# ldiff:   T - numerical diffusion on
# l2way:   T - feedback the nest domain information back to its mother domain
# lvgeo:   goestrophic wind
#          T - use large scale pgf
#
 cat << eof >> coamnl.$ddtg
 iadvct =     2,
 lspong =     t,
 rdtime =     240.0,
 nrdamp =     4,
 lddamp =     f,
 lralee =     f,
 robert =     0.2,
 ldiff  =     t,
 l2way  =     f,
 lvgeo  =     f,
eof

#
# physics options
#
# lcupar:  T - Kain-Fritsch convective parameterization on
# lcuppr:  T - Kain-Fritsch diagnostics print on
# icup :   3 - new KF
# lmoist:  T - allow for moisture
# lice:    T - ice physics on
# lgrpl:   T - graupel physics on
# lrad:    T - radiation on
# lflux:   T - surface fluxes on
# lsfcen:  T - allow surface fluxes and radiation to affect the ground t and q
#          idealized:  T- initialize pressure, or potential temperature, or
#                      add perturbation to horizontal wind
# dxmeso:  upper limit (in meter) of horizontal resolution below which the vertical
#          advection of tke, raindrops and snow are included in prognostic euq.
#
#

 cat << eof >> coamnl.$ddtg
 lcupar =     t,
 lcuppr =     f,
 icup   =     3,3,3,3,3,3,3,
 lmoist =     t,
 lice   =     t,
 lgrpl  =     t,
 lrad   =     t,
 lflux  =     t,
 lsfcen =     t,
 dxmeso = 10000.0,
eof

# surface options
#
# ilndflg:
1-D array, one for each domain
#          0 - user define land usage (in subroutine user_sfc.f)
#          1 - use lat-long database
# ilandu:  1-D array, one for each domain
#          0 - use climate landuse
#          1 - use USGS 1km landuse
# alndpct: if ilndflg=0 specify land percentage in the model.
#          also need to specify almin, albedo, alnnt, alpha
# almin:   minimum mixing length factor
# al0:     mixing length constant
# ialbflg: 1-D array, for each domain
#          0 - user define albedo (set albdo)
#          1 - use lat-long database (ignore albdo)
# albdo:   albedo
# iz0flg:  1-D array, for each domain
#          0 - user define surface roughness (set zrough)
#          1 - use lat-long database (ignore zrough)
# zrough:  surface roughness
# igwtflg: 1-D array, for each domain
#          0 - user define ground wetness (set sfcwet)
#          1 - use lat-long database (ignore sfcwet)
# sfcwet:  user define surface wetness
# iseaflg: 1-D array, defined for each domain
#          0 - user define sea surface temperature (set seatmp)
#          1 - use lat-long database (ignore seatmp)
# seatmp:  user define sea surface temperature
# itopoflg:1-D array, for each domain
#          0 - user define terrain; 1 - use lat-long database
# ltopoa:  T - read in and use terrain field
# nftopo:  number of passes through 3-point filter; used to smooth terrain
#          field
# cstline: coastline fudging factor (to define the coastline)
#          This number (between 0.0 and 1.0) vary depends on one's grid
#          resolution and area.  cstline=0.0 means the land-sea
#          value is not fudged.  A value close to 1.0 means you want to bring the
#          coastline more toward the land.
#
# type1k:  = 'DTED1_010' NIMA DTED1 1km terrain database
#          = 'DTED0_010' NIMA DTED0 1km terrain database
# lsilhout: T - compute terrain using the silhouette method
#           F - using the old COAMPS method terrain+(standard deviation*sdmult)
# nsrch:   1 D array (one for each grid) that define the size of the
#          box placed around a given grid point (no. of topo grids in the x and y
#          direction).
# topores: 1 D array (one for each grid) with value of 1.0 or 20.0.
#          topores=1.0 means using the 1 km topo data. topores=20.0 means using the
#          20 km topo data.
# iwvlng:  1 D array (one for each grid) with value of 2 or 4. iwvlng=4
#          means compute terrain on the 4 delx grid and interpolated
#          the 4 delx grid terrain field back to the COAMPS grid.
# silwgt:  1 D array (one for each grid) with value between 0 and 1.
#          silwgt=1 means fully weight the silhouette average otherwise
#          change the value of the topo estimate toward the average height
#          of all the points times silwgt.
# silmax:  1 D array (one for each grid) with value of 0 or 1.
#          silmax=1 means take the highest point in the search box
#          as the value of the topo. otherwise weight it as an average
#          of the max value and the silhouette value.
# nslflt:  1 D array (one for each grid) with value of 1 or 0.  nslflt=1
#          means filter the silhouette field.
#
 cat << eof >> coamnl.$ddtg
 ilandu   =    1,1,1,1,1,1,1,
 ilndflg  =    1,1,1,1,1,1,1,
 alndpct  =    1.,
 almin    =    0.02,
 al0      =    1.0,
 ialbflg  =    1,1,1,1,1,1,1,
 albdo    =    0.2,
 iz0flg   =    1,1,1,1,1,1,1,
 zrough   =    0.01,
 igwtflg  =    1,1,1,1,1,1,1,
 sfcwet   =    0.1,
 iseaflg  =    1,1,1,1,1,1,1,
 seatmp   =    285.,
 itopoflg =    1,1,1,1,1,1,1,
 ltopoa   =    t,
 nftopo   =    1,1,1,1,1,1,1,
 cstline  = 0.0,0.0,0.0,0.0,0.0,0.0,0.0,
 sdmult   = 1.0,
 lsilhout =  t,
 type1k   = 'DTED1_010',
eof

# Turbulence options
#
# ltke:    T - subgrid scale mixing on
# iashsgm: 1 - use deformation field for subgrid mixing
#          2 - use tke prediction
# iashsm:  1 - Ri dependent (Mellor & Yamada, 74)
#          2 - sm=sm0, sh=sh0
#          3 - sm=sm0, sh=(1.0+2.0 al2/al1)*sm0
#          4 - sm=sm0, sh=2.13*sm0
#          5 - Ri dependent (Mellor & Yamada, 82)
#
iamxgl:  mixing length identifier (1,2,3,4,5), see user's guide table D-1
# sh0:     constant; used to compute the tke coefficient sh
# sm0:     constant; used to compute the tke coefficient sm

 cat << eof >> coamnl.$ddtg
 ltke   =     t,
 iahsgm =     0,
 iashsm =     5,
 iamxgl  =    5,
 sh0    =     0.675,
```

```
 sm0    =     0.5,
eof

# objective analysis options
# loi:     T - multivariate optimum interpolation analysis
# loimf:   T - inner mesh multivariate optimum interpolation analysis
# lqanl:   T - pressure level specific humidity analysis using Cressman scheme
# ltanl:   T - pressure level temperature analysis using Cressman scheme
# lpseud:  T - output pseudo observation base on NOGAPS fields
# ibog1,jbog1:
#          If namelist parameter lpseud = true, then COAMPS creates pseudo-obs
#          for the MVOI. The size of the analysis volume for pseudo-obs is
#          controlled by namelist parameters ibog1 and jbog1.
#
  cat << eof >> coamnl.$ddtg
 loi    =     t,
 loimf  =     t,
 lqanl  =     t,
 ltanl  =     t,
 lpseud =     t,
 ivol   =     11,31,91,
 jvol   =     11,31,91,
 ibog1  =     11,
 jbog1  =     11,
eof

# Data assimilation options
#
# iupd:    data assimilation identifier
#          0 - none
#          1 - full update
#
2 - incremental update (it is recommended to use 2 for data assimilation)
# itauin:  frequency for obtaining and writing out coarse mesh boundary
#          tendencies from NOGAPS fields
# itauf:   end time (hr) for obtaining and writing out coarse mesh boundary
#          tendencies from NOGAPS fields
# icycle:  data assimilation cycle (hr)
# prbc:    NOGAPS pressure levels used to compute COAMPS bc

  cat << eof >> coamnl.$ddtg
 iupd   =     2,2,2,
 itaus  =    0, 0, 0,
 itauin =    6,0,0,
 itauf  =    24,0,0,
 icycle =    12,
 prbc   =     10.0,  20.0,  30.0,  50.0,  70.0, 100.0, 150.0
             200.0, 250.0, 300.0, 400.0, 500.0, 700.0, 800.0,
             850.0,900.0, 925.0, 950.0, 975.0,1000.0, 1013.20,
eof
#
# Input and output options
#
# ldiagt:
 tketotal and thetotal mpi diagnostic print
# kprnta:  print frequency for forecast fields (hr, min, sec)
# ifsave:  model output options
#          1 - specify save time
#          2 - specify save frequency (hr, min, sec)
# ksaves:  time (hr,min,sec) to save sigma data output
# isavefrq: output frequency (hr,min,sec) to save sigma data
# ksavef:  output frequency for writing model restart file (hr, min,sec)
#          it is also used to restart a model run at a nonzero time
# lprint:  T - more diagnostic prints
# lwritu:  read/write model output
#          T - unformatted (32-bit ieee)
#          F - formated
# ksavmsp: output frequency to compute and write out
#          the maximum lowest sigma wind speed. Defaults is -1,0,0.
# idbms:   flags to select the flat file type. Defaults is 1
# ksavdig: sigma output frequency (hr,min,sec) for the forward and
#          backward digital runs
# ksavpcp: frequency (h,m,s) to tip the precipitation bucket
#
  cat << eof >> coamnl.$ddtg
 ldiagt = t,
 kprnta = 9999,  0,  0,
 ifsave = 2,
 ksaves =    1, 0,  0,
              2, 0,  0,
 ksavpcp =  12, 0,  0,
            12, 0,  0,
 isavefrq = 12, 0,  0,
 ksavea =   25, 56,  0,
 ksavdig=    1, 0,  0,
 ksavmsp=   -1, 0,  0,
 idbms  =    4,
 lprint =    f,
 lwritu =    t,
 &end
eof
#
# datfil:   directory path for ADP data
# dsobsw:   directory path of output observations
#

  cat << eof >> mvoinl.$ddtg
 &mvoinl
  datfil='$scrDir/adp$ddtg/',
  dsobsw='$dataDir/',
 &end

eof

#
# kcctyp: integer.  1, output at specified time
#                   2, output at a frequency
#
# kccfrq(3):  integer array.  frequency of output (hr,min,sec)
#
# kcctime(3,kcctot): integer array. specify time of output (hr,min,sec)
#           kcctot=50
#
#
# nadvtyp: integer. type of advection scheme used for transport
#          ntpadv=1, upstream advection with cubic-spline interpolation
#                            of monotonicity and positiveness
#          ntpadv=2, Bott's 5th-order flux-form advection
#                            with area-preservation and positiveness
#
# emssflg: logical. true, have source emission or mass release
#                   false, no emission
#
# nintcc:  integer.  = 0, zero initial concentration field
#                    = 1, user provides initial concentration field
#
# irrgsrc: integer. = 0, regular source emission strength setup
#                       given nsrc,cntlat,cntlon,cnthgt,emstrng, etc.
#                   = 1, user provides any irregular or
#                        real case emission distribution
#
# nsrc: integer.  number of source emission sites (default=1 & max=10)
#                       (nsrc = or > 1, used with irrgsrc=0)
#
# cntlat (nsrc): real array.  latitude of emission centers (irrgsrc=0)
# cntlon (nsrc): real array.  longitude of emission centers
#
# cnthgt (nsrc): real array. height (m) of emission centers above ground
#               (default=10m,i.e. first model level; used with irrgsrc=0)
#
# ntems: integer. number of emission changes in time for instantaneous
#                 and continuous releases. (used for both irrgsrc=0/1)
#                 ntems = 1, constant emission (ntems = or > 1)
#
# emstrng (ntems,nsrc): real array.  emission strength or fluxes
#                               (kg/delt/volume)
#                               at each time and at each site (irrgsrc=0)
#
# emstime(3,ntems): integer array. time (hr,min,sec) when emission changes,
#                               from the beginning of the forecast
#                  (for instantaneous release, the next time for zero mass
#                   release should be < or = one delt(nestcc)
#                   to make sure the release stops at next delt )
#                  (used for both irrgsrc=0/1)
#
# ngrdems:  integer. ngrdems=1, emission from single grid volume
#                    ngrdems=2, emission from multiple grid volumes
#                           (oval cloud shape) (irrgsrc=0)
#
# hhrad (nsrc): real array. horizontal radius (m) of cloud-shape volume
#                       for each site (used with ngrdems=2)
# zzrad (nsrc): real array. vertical radius (m) of cloud-shape volume
#                       for each site (used with ngrdems=2)
#                           (used for both irrgsrc=0/1)
#
# ccycle: logical.  true, warm start i.e. using previous forecast
#                   false, cold start (default is true )
#
# aerotme0 (3): integer array (hr,min,sec).
#               beginning time of tracer simulation within the first run
#                           [aerotme0 < COAMPS forecast time]
#
# lmbudg: logical.  flag to control mass budget calculation
#                   lmbudg=true, turn on; lmbudg=false, off
#
################################################################################
  cat << eof
>> aeronl.$ddtg
 &aeronl
 kcctyp = 2,
 kccfrq =   1, 0, 0,
 nadvtyp = 2,
 cntlat  = 30.75, 30.75, 30.75,
 cntlon  = 46.45, 46.45, 46.45,
 cnthgt  =   10., 30., 55.,
 ntems   = 2
```

```
 emstrng = 200., 0.,
           200., 0.,
            300., 0.,
 emstime =   1, 0, 0,
             1, 0, 1,
 ngrdems =   1,
 ccycle  =   f,
 aerotme0 = 1, 0, 0,
 &end
eof

#####################################################################
#####################################################################
#   end user modification
#####################################################################

#####################################################################
#####################################################################
#   set analysis and forecast namelists
#####################################################################
#####################################################################

    cat gridnl.$ddtg atmosnl.$ddtg dsetnl.$ddtg coamnl.$ddtg mvoinl.$ddtg >>
anal.$ddtg.$expname
    cat gridnl.$ddtg atmosnl.$ddtg dsetnl.$ddtg coamnl.$ddtg aeronl.$ddtg >> fcst.$ddtg.$expname
    rm -f gridnl.$ddtg atmosnl.$ddtg dsetnl.$ddtg coamnl.$ddtg mvoinl.$ddtg aeronl.$ddtg

#####################################################################
#####################################################################
#   set forecast date environment parameters
#####################################################################
#####################################################################

  CRDATE=$ddtg
  export CRDATE

#####################################################################
#####################################################################
#   make sure the coamps scratch data directory exists
#####################################################################
#####################################################################
  if [ ! -d $dataDir ]; then mkdir -p $dataDir; fi;
  if [ ! -d $dataDir/forward ]; then mkdir -p $dataDir/forward; fi;
  if [ ! -d $dataDir/backward ]; then mkdir -p $dataDir/backward; fi;
#####################################################################
#####################################################################
#   starts with clean directory if running analysis
#####################################################################
#####################################################################
  if [ $def2 = "1" -o $def2 = "3" ]; then
    echo "clean out $dataDir"
    rm -f -r $dataDir
    mkdir -p $dataDir
  fi
#####################################################################
#####################################################################
#   make sure the COAMPS history data directory exists
#####################################################################
#####################################################################

  if [ ! -d $saveDir ]; then mkdir -p $saveDir; fi;

  if [ $get_coamps = "yes" ]; then
    echo "get $def4 hour old coamps data"
    if [ -s $saveDir/${ddtgm1}_iupd.tar ]; then
      cd $dataDir
      tar xvf $saveDir/${ddtgm1}_iupd.tar
      tar xvf $saveDir/${ddtgm1}_stats.tar
    fi
    if [ -s $saveDir/${ddtgm1}_coda.tar ]; then
      cd $dataDir
      tar xvf $saveDir/${ddtgm1}_coda.tar
    fi
  fi

#####################################################################
#####################################################################
#   make sure the executable is up to date
#####################################################################
#####################################################################

  if [ $recompile = "yes" ]; then
     cd $modDir
     echo "make $osname"
     make $osname
     if [ $? -ne 0 ]
      then
      echo "$problem in compiling!  coamps $expname run stops"
      exit
     fi
  fi

#####################################################################
#####################################################################
#   copy nogaps, adp, and ocnqc files to the scratch data directory
#####################################################################
#####################################################################

    if [ $get_nogaps = "yes" -a -s $nogapsDir/f$ddtg.tar ]; then
      echo " get nogaps data $nogapsDir/f$ddtg.tar"
      cd $scrDir
      tar xf $nogapsDir/f$ddtg.tar
    fi

    if [ $get_adp = "yes" -a -s $adpDir/adp$ddtg.tar.Z ]; then
      echo " get adp data $adpDir/adp$ddtg.tar.Z"
      cd $scrDir
      if [ ! -d adp$ddtg ]; then mkdir adp$ddtg; fi
      cd adp$ddtg
      cp $adpDir/adp$ddtg.tar.Z .
      uncompress adp$ddtg.tar.Z
      tar xf adp$ddtg.tar
      rm -f adp$ddtg.tar
    fi

    if [ $get_ocn = "yes" ]; then
      echo "get ocnqc data"
      cd $scrDir
      if [ ! -d  ocnqc/altim ]; then mkdir -p ocnqc/altim; fi;
      if [ ! -d  ocnqc/mcsst ]; then mkdir -p ocnqc/mcsst; fi;
      if [ ! -d  ocnqc/profile ]; then mkdir -p ocnqc/profile; fi;
      if [ ! -d  ocnqc/ship ]; then mkdir -p ocnqc/ship; fi;
      if [ ! -d  ocnqc/ssmi ]; then mkdir -p ocnqc/ssmi; fi;

#
#  search for previous 4 days mcsst, ship, and ssmi data
#
      while [ $ocnddtg -ge $ocnddtgm1 -a $ocnddtg -le $ocnddtgp1 ]
      do
        if [ -s $ocnDir/mcsst/$ocnddtg.mcsst.Z ]; then
          cd $scrDir/ocnqc/mcsst
          if [ ! -s $ocnddtg.mcsst.Z ]; then
            cp $ocnDir/mcsst/$ocnddtg.mcsst.Z .
            uncompress $ocnddtg.mcsst.Z
          fi
        fi

        if [ -s $ocnDir/ship/$ocnddtg.ship.Z ]; then
          cd $scrDir/ocnqc/ship
          if [ ! -s $ocnddtg.ship.Z ]; then
            cp $ocnDir/ship/$ocnddtg.ship.Z .
            uncompress $ocnddtg.ship.Z
          fi
        fi

        if [ -s $ocnDir/ssmi/$ocnddtg.ssmi.Z ]; then
          cd $scrDir/ocnqc/ssmi
          if [ ! -s $ocnddtg.ssmi.Z ]; then
            cp $ocnDir/ssmi/$ocnddtg.ssmi.Z .
            uncompress $ocnddtg.ssmi.Z
          fi
        fi
        ocnddtg=`echo $ocnddtg 24 | $newdtg`
      done
    fi

#####################################################################
#####################################################################
#   execute the analysis
#####################################################################
#####################################################################

  if [ $def2 = "1" -o $def2 = "3" ]; then

    cd $runDir
    echo " run coamps analysis "
    date > start.time
    if [ -s namelist ];then
      rm -f namelist
    fi
    cp anal.$ddtg.$expname namelist
    cp oanl.$ddtg oanl
    echo " $analysis"
#   ( time 2> time.a $analysis ) > log.a.$expname.$ddtg
    time 2> time.a $analysis > log.a.$expname.$ddtg

    if [ $? -ne 0 ]; then
      echo "$analysis crashed!  coamps $expname run stops"
      exit
    else
      date > end.time

echo "starting time of coampa anlysis:" >> log.a.$expname.$ddtg
      cat start.time >> log.a.$expname.$ddtg
      echo "ending time of coamps analysis:" >> log.a.$expname.$ddtg
```

```
      cat end.time >> log.a.$expname.$ddtg
      cat time.a >> log.a.$expname.$ddtg
#########################################################################
#########################################################################
#     save the initial conditions
#########################################################################
#########################################################################
      if [ $save_data = "yes" ]; then
        cd $dataDir
        tar cvf $saveDir/${ddtg}_coda.tar *1o* *2o*
        tar cvf $saveDir/${ddtg}_init.tar bd*sig*$ddtg* data*${ddtg}* \
        *_sfc*${ddtg}_00000000* *pre*${ddtg}_00000000* \
        *_zht*${ddtg}_00000000* *${ddtgm1}_00$def4*
      fi
    fi
    rm -f start.time end.time time.a
    echo " "
    echo " ------------------------------"
    echo " analysis complete ....."
    echo " ------------------------------"
    echo " "
  fi

#########################################################################
#########################################################################
#   execute the forecast model
#########################################################################
#########################################################################

  if [ $def2 = "2" -o $def2 = "3" ]
    then
    cd $runDir
    echo " run coamps forecast"
    date > start.time
    if [ -s namelist ];then
      rm -f namelist
    fi
    cp fcst.$ddtg.$expname namelist

#########################################################################
# execute parallel forecast job
#########################################################################

#       echo " $forecast > log.m.${nproc}p.$ddtg"
#       ( time 2> time.m $forecast ) > log.m.${nproc}p.$ddtg
        time 2> time.m $forecast > log.m.${nproc}p.$ddtg

      if [ $? -ne 0 ]
       then
       echo "$forecast crashed!  comaps $expname run stops"
       if [ $osname = "alpha" ];then
         mpiclean
       elif [ $osname = "linux" ]; then
         lamclean
       fi
       exit
      else
       date > end.time
       echo "starting time of forecast model:" >> log.m.${nproc}p.$ddtg
       cat start.time >> log.m.${nproc}p.$ddtg
       echo "ending time of forecast model:" >> log.m.${nproc}p.$ddtg
       cat end.time >> log.m.${nproc}p.$ddtg
       cat time.m >> log.m.${nproc}p.$ddtg
#########################################################################
#########################################################################
#     save the model forecast
#########################################################################
#########################################################################
      if [ $save_data = "yes" ]; then
        cd $dataDir
        tar cvf $saveDir/${ddtg}_sgl.tar data*$ddtg* *_sig*$ddtg*fcstfld
        tar cvf $saveDir/${ddtg}_stats.tar data*$ddtg* *_pre*$ddtg*0012*
        tar rvf $saveDir/${ddtg}_stats.tar *_pre*$ddtg*0024* *_sfc*$ddtg* *_zht*$ddtg*
        tar cvf $saveDir/${ddtg}_iupd.tar data*$ddtg* slp*$ddtg* \
        *_sig*$ddtg*00${def4}* *_sfc*$ddtg*00${def4}* *_zht*$ddtg*00${def4}*
terr*${ddtg}*
      fi
    fi
    rm -f start.time end.time time.m
    echo " "
    echo " ---------------------------------"
    echo " forecast complete ....."
    echo " ------------------------------"
    echo " "
  fi

#########################################################################
#########################################################################
#   execute the coamps plotxy
#########################################################################
#########################################################################
  if [ $run_plotxy = "yes" ]; then
    echo "run plotxy"
    cd $runDir
    if [ -s plotxy.sh ]; then
      chmod +x plotxy.sh
      plotxy.sh $ddtg $dataDir $PLOTXY ocards.plotxy $expname
      mv gmeta $saveDir/gmeta.$ddtg
    else
      echo " plotxy script plotxh.sh not found in directory $runDir"
    fi
  fi

#########################################################################
#########################################################################
#   clean up unwanted files
#########################################################################
#########################################################################

  cd $runDir
  rm -f coda.*obs xgetgg* fort.21

#########################################################################
#########################################################################
#   job is complete
#########################################################################
#########################################################################
#
  if [ $def2 = "1" -o $def2 = "3" ]; then
   echo "please check your coamps analysis log file log.a.$expname.$ddtg"
  fi

  if [ $def2 = "2" -o $def2 = "3" ]; then
   echo "please check your coamps forecast log file log.m.${NP}p.$ddtg"
  fi

   echo " "
   echo "======================================================="

#########################################################################
#  submit the next job
#########################################################################
   if [ $def1 -lt $def3 ];then
     cd $runDir
     $runscript $ddtgp1 $def2 $def3 $def4
   fi

EOF
#
# execute the coamps run script
#
chmod +x co.$1.sh
if [ $Interactive = "yes" ]; then
  coamps.sh
else
  $batch co.$1.sh
  echo "$batch co.$1.sh"
  rm -f co.$1.sh
fi
echo "coamps run completed"


#########################################################################
#  end of coamps run script
#########################################################################
```

Appendix E

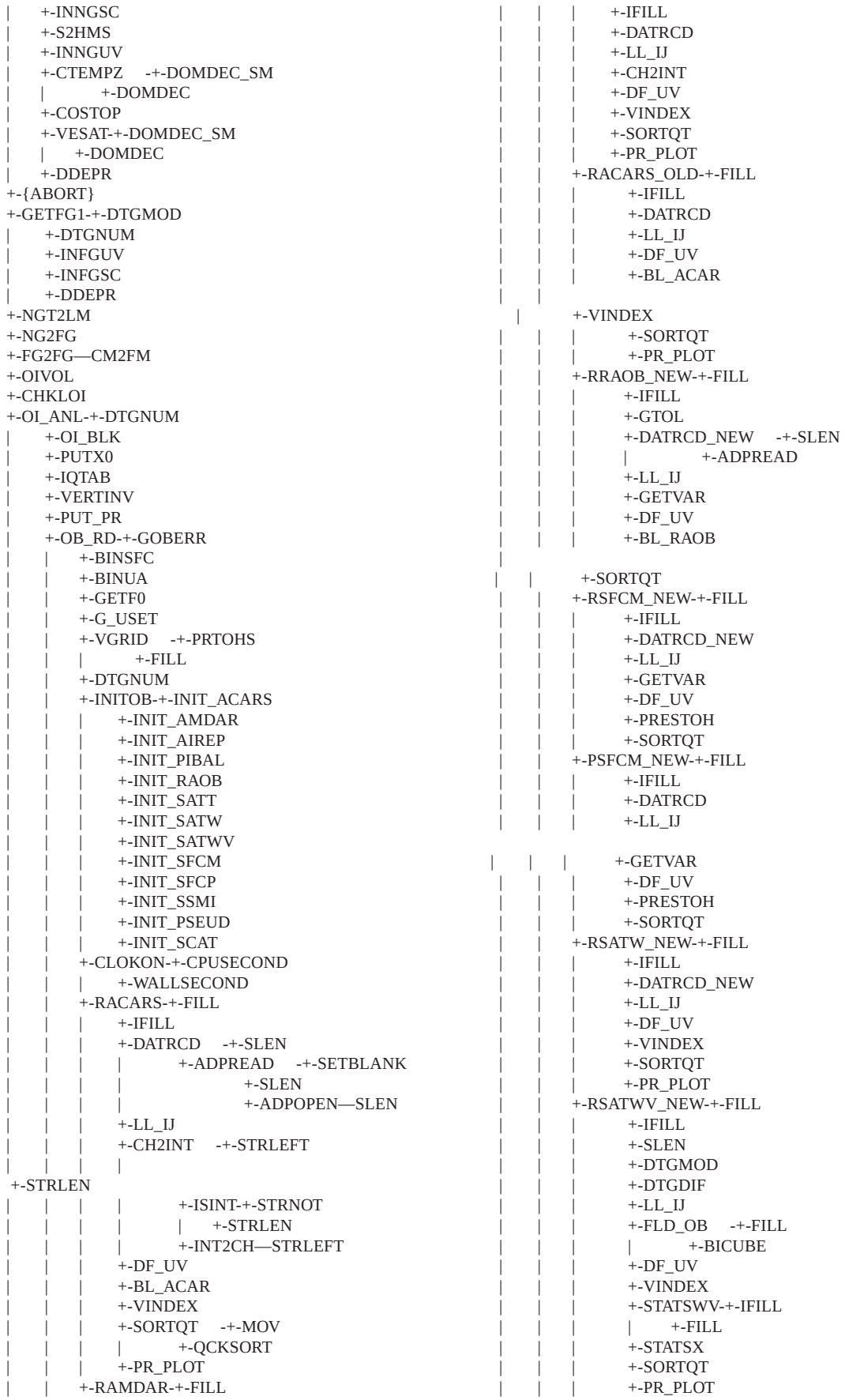
COAMPS Atmosphere Analysis Flow Chart

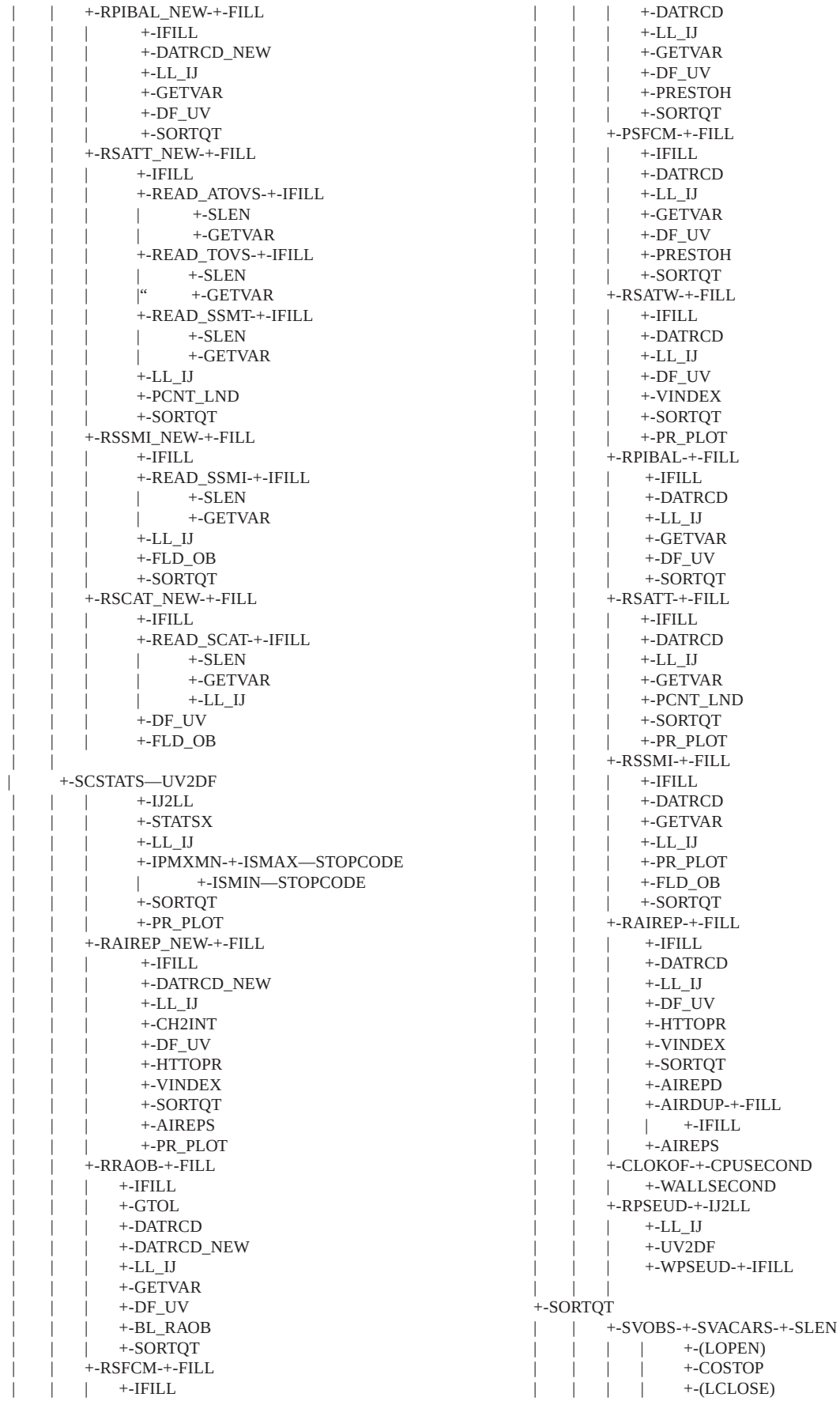
This is a primary tree starting at the program ‘ATMOS_ANALYSIS’.

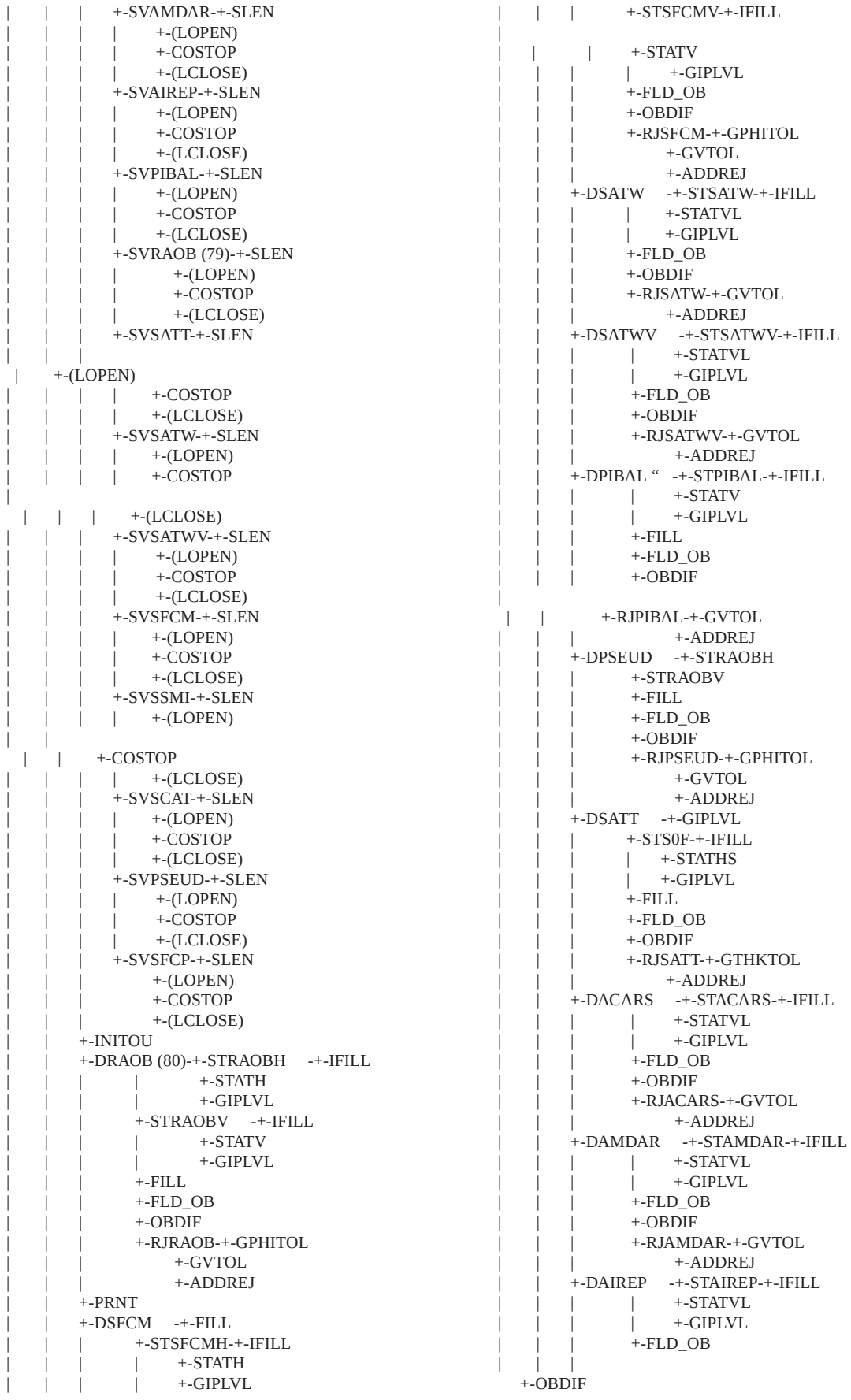
```
ATMOS_ANALYSIS
+-DTGOPS-+-{PXFGETENV}
|        +-{GETENV}
|        +-DTGCHK
|        +-{TIME}
|        +-{GMTIME}
|        +-DTGMOD-+-DTGNUM—DTGCHK
|                 +-DTGYRHR
+-DOMDEC_SM_INIT
+-MEMA
+-COAMA
 +-DOMDEC_SM
 +-CPUSECOND-+-{SECOND}
 |           +-{ETIME}
 +-WALLSECOND—{GET_WALLT}
 +-NSTLVLS-+-DOMDEC
 |         +-STOPCODE
 |
 +-CHEKMN-+-DOMDEC_SM
 |        +-DOMDEC
 |        +-COSTOP-+-DOMDEC_SM
 |        |        +-(DBSTOP)
 |        |        +-STOPCODE
 |        +-STOPCODE
 +-CHECKZ-+-DOMDEC_SM
 |        +-DOMDEC
 +-CHKISIS-+-(GGRD)
 |         +-COSTOP
 |         +-(GETGEOM)
 |         +-IJ2LL
 +-ESAT1
 +-COSTOP
 +-BCINDX—COSTOP
 +-GEOSTD
 +-GRDIJ-+-DOMDEC_SM
 +-GRID
 +-CHEKTS-+-DOMDEC_SM
 +-HM2UV-+-DOMDEC_SM
 +-QPRNTN
 +-STAGF
 +-XYN-+-DOMDEC_SM
 +-GCORNER-+-CONVLONG
 |         +-MINMAX
 +-WDATA    -+-S2HMS
 |           +-(GLWR)
 |           +-COSTOP
 |           +-SLEN
 |           +-DFALTS
 |           +-DATAW-+-S2HMS
 |           |       +-(ASSIGN)
 |           |       +-(LOPEN)
 |           |       +-(LCLOSE)
 |           +-DATAW_NEW-+-S2HMS
 |           |           +-(ASSIGN)
 |           |           +-(LOPEN)
 |           |           +-(LCLOSE)
 |
 |           +-{ABORT}
 +-DTGMOD
 +-RDATA    -+-S2HMS
 |           +-(GLRD)
 |           +-SLEN
 |           +-DFALTS
 |           +-DATAR-+-S2HMS
 |           |       +-(ASSIGN)
 |           +-DATAR_NEW-+-S2HMS
 |           |           +-(ASSIGN)
 |           +-{ABORT}
 +-CHECKGEOG
 +-SLEN
 +-S2HMS
 +-CODA (see ocean analysis flow chart)
 |
 +-CM2FM    —BCUBIC    -+-BCINIT
 |                      +-BCTDG1
 |                      +-BCTDG2
 |                      +-BCTDX1
 +-FG2CM-+-LL2IJ
 |       +-LIJSG
 +-SFCPAR
 |+-SLEN
 |+-DTGNUM
 |+-DTGMOD
 |+-RDATA
 |+-LANDSEA    —LLSEA-+-RIGHT_SHIFT-+-(SHIFTR)
 ||                   |              +-(RSHFT)
 ||                   +-GETLS-+-(ASSIGN)
 ||                   |       +-(GEUSRCXT)
 ||                   |       +-(GEUSRRD)
 ||                   +-GENT-+-RIGHT_SHIFT
 ||                          +-LEFT_SHIFT-+-(SHIFTL)
 ||                                       +-(LSHFT)
 |+-FILT25
 |+-(GGRD)
 |+-TOPODEV    —TOPO1KM-+-GGSILGET    —SLEN
 ||                    +-SLEN
 ||                    +-SILINTRP
 ||                    +-GGPTGET    -+-(PARMTAB)
 ||                                  +-STRLEN
 ||                                  +-(TYPETAB)
 ||                                  +-(NRESTAB)
 ||                                  +-(RESTAB)
 ||                                  +-(XGETGG)
 |+-TOPOSIL    -+-GRDIJ_MOVE
 ||            +-GRID_MOVE
 ||            +-TSIL1KM-+-LANDSEA
```

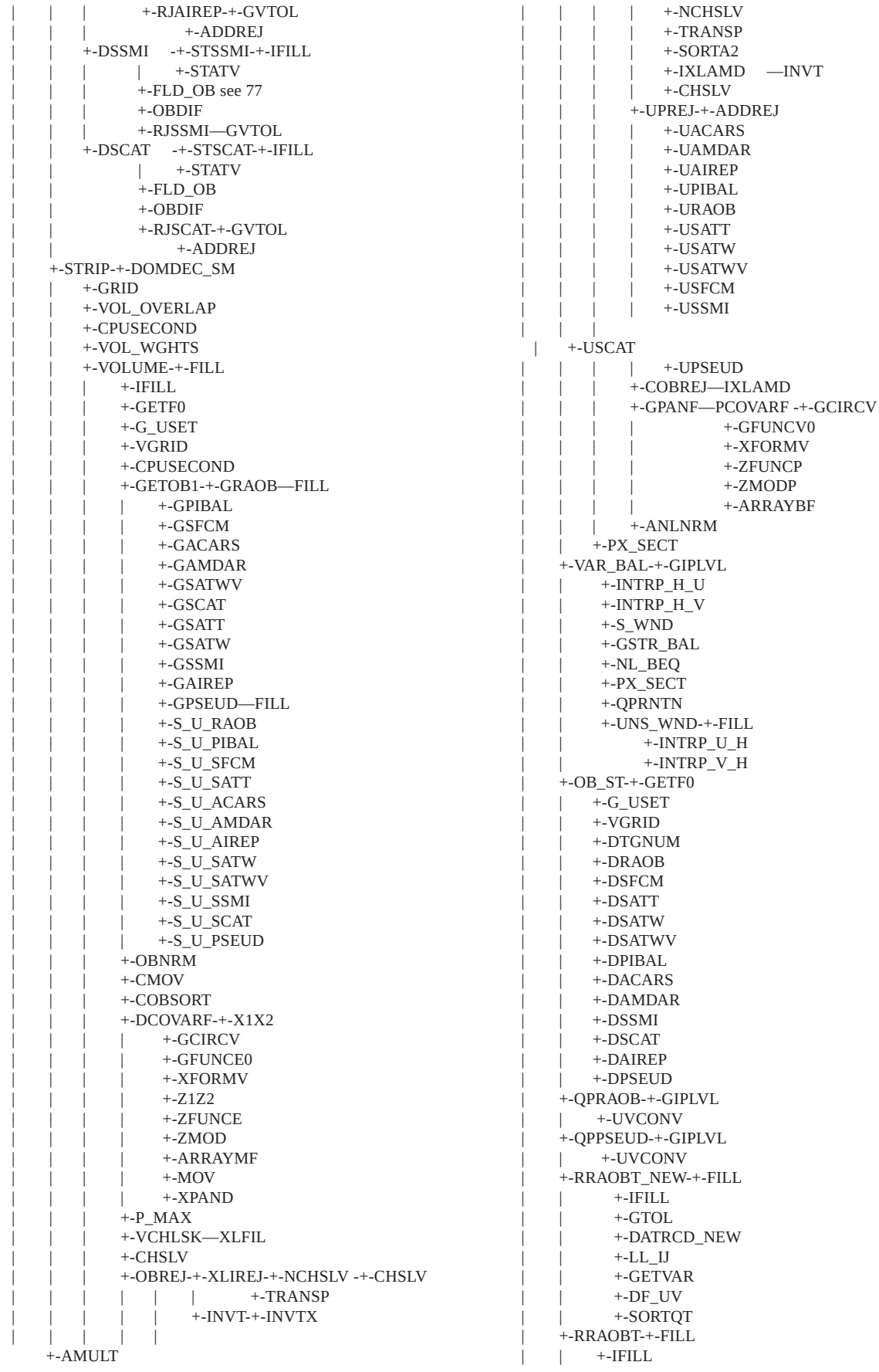
```
||          |       +-SLEN
||          |       +-GGSILGET
||          |       +-GRIDBOX
||          |       +-SILINTRP
||          |       +-GGPTGET
||          |       +-FILT25
||          |       +-BINT    —ONED
||          +-TSIL20KM-+-GRIDBOX
||                     +-FILT25
||                     +-BINT
|+-CLIMIN    —(ASSIGN)
|+-(ASSIGN)
|+-LANDUS-+-DOMDEC_SM
||      +-GRID
||      +-SLEN
||      +-(ASSIGN)
|+-QPRNTN
+-THIRDP
+-REMV1P
+-GRDIJ_MOVE
+-GRID_MOVE
+-SFCPAR_MOVE-+-SLEN
|          +-DTGNUM
|          +-DTGMOD
|          +-RDATA
|          +-LANDSEA
|          +-FILT25
|          +-(GGRD)
|          +-TOPODEV
|          +-TOPOSIL
|          +-(ASSIGN)
+-GETSST-+-DTGMOD
|      +-DTGNUM
|      +-INNGSC    -+-S2HMS
|      |           +-RDATA
|      |           +-BINT
|      |           +-BCUBIC
|      +-INFGSC    —RDATA
+-USER_SFC
+-TWOWAY
+-TMATCH
+-TMATCH_M
+-FMNN    -+-DOMDEC_SM
|         +-DOMDEC
+-ATOPO    -+-DOMDEC_SM
|          +-DOMDEC
+-SOILTP-+-SLEN
|      +-CLIMIN
|      +-DTGNUM
|      +-GEOSTD
+-ASTATE-+-DOMDEC_SM
|      +-DOMDEC
|      +-PSTD
|      +-TSTD
+-REFSND-+-DOMDEC_SM
|      +-DOMDEC
|      +-ESAT1
|      +-RS
|      +-STOPCODE
+-ICM2FG-+-ESAT1
|      +-CM2FM
|      +-ESATV
|      +-QPRNTN
+-ISTATE-+-DOMDEC_SM
|      +-DOMDEC
|      +-ESAT1
|      +-INTZ
|      +-FLDS08-+-ESAT1
|      |        +-TH2T
|      |        +-ESATV
|      +-FLDS03
|      +-FLDS13-+-DOMDEC
|      |        +-(MPI_BCAST)
|      +-ESATV
|      +-QPRNTN
+-SNDPRNT
+-IOSFC-+-WDATA
|      +-RDATA
+-IOANL-+-IOZAVG-+-S2HMS
|      |          +-SLEN
|      |          +-STOPCODE
|
 +-IOSFC0-+-S2HMS
|      |          +-SLEN
|      |          +-(LOPEN)
|      |          +-COSTOP
|      |          +-(LCLOSE)
|      +-IOSFCT-+-S2HMS
|      |          +-SLEN
|      |          +-(LOPEN)
|      |          +-COSTOP
|
|      +-(LCLOSE)
|      +-IOATMT-+-S2HMS
|      |          +-SLEN
|      |          +-COSTOP
|      |          +-STOPCODE
|      +-FMNN
|      +-ATOPO
|      +-INDXBD    —DOMDEC
+-READOC-+-DOMDEC_SM
|      +-DOMDEC
|
+-SLEN
|      +-OCORD
|      +-COSTOP
|      +-STOPCODE
|      +-DFALTS
|      +-SORTO
+-READO-+-DOMDEC_SM
|      +-DOMDEC
|      +-SLEN
|      +-DFALTS
|      +-STOPCODE
|      +-RMDUPO
|      +-SORTO
+-LASTO

+-INDXBD
+-GETHTY-+-DTGMOD
|      +-DTGNUM
|      +-INNGSC
|      +-INFGSC
|      +-INNGUV    -+-S2HMS
|      |           +-RDATA
|      |           +-BCUBIC
|      |           +-UV2DF
|      |           +-UVGRID
|      +-INFGUV    —RDATA
+-OUTHTY—WDATA
+-NETHTY-+-DTGMOD
|      +-DTGNUM
|      +-BINT
|      +-INFGSC
|      +-CM2FM
|      +-INFGUV
+-GETNG-+-DTGMOD
|      +-DTGNUM
```


+-INNGSC
+-S2HMS
+-INNGUV
+-CTEMPZ -+-DOMDEC_SM
+-DOMDEC
+-COSTOP
+-VESAT-+-DOMDEC_SM
+-DOMDEC
+-DDEPR
+-{ABORT}
+-GETFG1-+-DTGMOD
+-DTGNUM
+-INFGUV
+-INFGSC
+-DDEPR
+-NGT2LM
+-NG2FG
+-FG2FG—CM2FM
+-OIVOL
+-CHKLOI
+-OI_ANL-+-DTGNUM
+-OI_BLK
+-PUTX0
+-IQTAB
+-VERTINV
+-PUT_PR
+-OB_RD-+-GOBERR
+-BINSFC
+-BINUA
+-GETF0
+-G_USET
+-VGRID -+-PRTOHS
+-FILL
+-DTGNUM
+-INITOB-+-INIT_ACARS
+-INIT_AMDAR
+-INIT_AIREP
+-INIT_PIBAL
+-INIT_RAOB
+-INIT_SATT
+-INIT_SATW
+-INIT_SATWV
+-INIT_SFCM
+-INIT_SFCP
+-INIT_SSMI
+-INIT_PSEUD
+-INIT_SCAT
+-CLOKON-+-CPUSECOND
+-WALLSECOND
+-RACARS-+-FILL
+-IFILL
+-DATRCD -+-SLEN
+-ADPREAD -+-SETBLANK
+-SLEN
+-ADPOPEN—SLEN
+-LL_IJ
+-CH2INT -+-STRLEFT
+-STRLEN
+-ISINT-+-STRNOT
+-STRLEN
+-INT2CH—STRLEFT
+-DF_UV
+-BL_ACAR
+-VINDEX
+-SORTQT -+-MOV
+-QCKSORT
+-PR_PLOT
+-RAMDAR-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-CH2INT
+-DF_UV
+-VINDEX
+-SORTQT
+-PR_PLOT
+-RACARS_OLD-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-DF_UV
+-BL_ACAR
+-VINDEX
+-SORTQT
+-PR_PLOT
+-RRAOB_NEW-+-FILL
+-IFILL
+-GTOL
+-DATRCD_NEW -+-SLEN
+-ADPREAD
+-LL_IJ
+-GETVAR
+-DF_UV
+-BL_RAOB
+-SORTQT
+-RSFCM_NEW-+-FILL
+-IFILL
+-DATRCD_NEW
+-LL_IJ
+-GETVAR
+-DF_UV
+-PRESTOH
+-SORTQT
+-PSFCM_NEW-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-GETVAR
+-DF_UV
+-PRESTOH
+-SORTQT
+-RSATW_NEW-+-FILL
+-IFILL
+-DATRCD_NEW
+-LL_IJ
+-DF_UV
+-VINDEX
+-SORTQT
+-PR_PLOT
+-RSATWV_NEW-+-FILL
+-IFILL
+-SLEN
+-DTGMOD
+-DTGDIF
+-LL_IJ
+-FLD_OB -+-FILL
+-BICUBE
+-DF_UV
+-VINDEX
+-STATSWV-+-IFILL
+-FILL
+-STATSX
+-SORTQT
+-PR_PLOT


+-RPIBAL_NEW-+-FILL
+-IFILL
+-DATRCD_NEW
+-LL_IJ
+-GETVAR
+-DF_UV
+-SORTQT
+-RSATT_NEW-+-FILL
+-IFILL
+-READ_ATOVS-+-IFILL
+-SLEN
+-GETVAR
+-READ_TOVS-+-IFILL
+-SLEN
+-GETVAR
+-READ_SSMT-+-IFILL
+-SLEN
+-GETVAR
+-LL_IJ
+-PCNT_LND
+-SORTQT
+-RSSMI_NEW-+-FILL
+-IFILL
+-READ_SSMI-+-IFILL
+-SLEN
+-GETVAR
+-LL_IJ
+-FLD_OB
+-SORTQT
+-RSCAT_NEW-+-FILL
+-IFILL
+-READ_SCAT-+-IFILL
+-SLEN
+-GETVAR
+-LL_IJ
+-DF_UV
+-FLD_OB
+-SCSTATS—UV2DF
+-IJ2LL
+-STATSX
+-LL_IJ
+-IPMXMN-+-ISMAX—STOPCODE
+-ISMIN—STOPCODE
+-SORTQT
+-PR_PLOT
+-RAIREP_NEW-+-FILL
+-IFILL
+-DATRCD_NEW
+-LL_IJ
+-CH2INT
+-DF_UV
+-HTTOPR
+-VINDEX
+-SORTQT
+-AIREPS
+-PR_PLOT
+-RRAOB-+-FILL
+-IFILL
+-GTOL
+-DATRCD
+-DATRCD_NEW
+-LL_IJ
+-GETVAR
+-DF_UV
+-BL_RAOB
+-SORTQT
+-RSFCM-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-GETVAR
+-DF_UV
+-PRESTOH
+-SORTQT
+-PSFCM-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-GETVAR
+-DF_UV
+-PRESTOH
+-SORTQT
+-RSATW-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-DF_UV
+-VINDEX
+-SORTQT
+-PR_PLOT
+-RPIBAL-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-GETVAR
+-DF_UV
+-SORTQT
+-RSATT-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-GETVAR
+-PCNT_LND
+-SORTQT
+-PR_PLOT
+-RSSMI-+-FILL
+-IFILL
+-DATRCD
+-GETVAR
+-LL_IJ
+-PR_PLOT
+-FLD_OB
+-SORTQT
+-RAIREP-+-FILL
+-IFILL
+-DATRCD
+-LL_IJ
+-DF_UV
+-HTTOPR
+-VINDEX
+-SORTQT
+-AIREPD
+-AIRDUP-+-FILL
+-IFILL
+-AIREPS
+-CLOKOF-+-CPUSECOND
+-WALLSECOND
+-RPSEUD-+-IJ2LL
+-LL_IJ
+-UV2DF
+-WPSEUD-+-IFILL
+-SORTQT
+-SVOBS-+-SVACARS-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)


+-SVAMDAR-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVAIREP-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVPIBAL-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVRAOB (79)-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVSATT-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVSATWV-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVSFCM-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVSSMI-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVSCAT-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVPSEUD-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-SVSFCP-+-SLEN
+-(LOPEN)
+-COSTOP
+-(LCLOSE)
+-INITOU
+-DRAOB (80)-+-STRAOBH -+-IFILL
+-STATH
+-GIPLVL
+-STRAOBV -+-IFILL
+-STATV
+-GIPLVL
+-FILL
+-FLD_OB
+-OBDIF
+-RJRAOB-+-GPHITOL
+-GVTOL
+-ADDREJ
+-PRNT
+-DSFCM -+-FILL
+-STSFCMH-+-IFILL
+-STATH
+-GIPLVL
+-STSFCMV-+-IFILL
+-STATV
+-GIPLVL
+-FLD_OB
+-OBDIF
+-RJSFCM-+-GPHITOL
+-GVTOL
+-ADDREJ
+-DSATW -+-STSATW-+-IFILL
+-STATVL
+-GIPLVL
+-FLD_OB
+-OBDIF
+-RJSATW-+-GVTOL
+-ADDREJ
+-DSATWV -+-STSATWV-+-IFILL
+-STATVL
+-GIPLVL
+-FLD_OB
+-OBDIF
+-RJSATWV-+-GVTOL
+-ADDREJ
+-DPIBAL “ -+-STPIBAL-+-IFILL
+-STATV
+-GIPLVL
+-FILL
+-FLD_OB
+-OBDIF
+-RJPIBAL-+-GVTOL
+-ADDREJ
+-DPSEUD -+-STRAOBH
+-STRAOBV
+-FILL
+-FLD_OB
+-OBDIF
+-RJPSEUD-+-GPHITOL
+-GVTOL
+-ADDREJ
+-DSATT -+-GIPLVL
+-STS0F-+-IFILL
+-STATHS
+-GIPLVL
+-FILL
+-FLD_OB
+-OBDIF
+-RJSATT-+-GTHKTOL
+-ADDREJ
+-DACARS -+-STACARS-+-IFILL
+-STATVL
+-GIPLVL
+-FLD_OB
+-OBDIF
+-RJACARS-+-GVTOL
+-ADDREJ
+-DAMDAR -+-STAMDAR-+-IFILL
+-STATVL
+-GIPLVL
+-FLD_OB
+-OBDIF
+-RJAMDAR-+-GVTOL
+-ADDREJ
+-DAIREP -+-STAIREP-+-IFILL
+-STATVL
+-GIPLVL
+-FLD_OB
+-OBDIF


+-RJAIREP-+-GVTOL
+-ADDREJ
+-DSSMI -+-STSSMI-+-IFILL
+-STATV
+-FLD_OB see 77
+-OBDIF
+-RJSSMI—GVTOL
+-DSCAT -+-STSCAT-+-IFILL
+-STATV
+-FLD_OB
+-OBDIF
+-RJSCAT-+-GVTOL
+-ADDREJ
+-STRIP-+-DOMDEC_SM
+-GRID
+-VOL_OVERLAP
+-CPUSECOND
+-VOL_WGHTS
+-VOLUME-+-FILL
+-IFILL
+-GETF0
+-G_USET
+-VGRID
+-CPUSECOND
+-GETOB1-+-GRAOB—FILL
+-GPIBAL
+-GSFCM
+-GACARS
+-GAMDAR
+-GSATWV
+-GSCAT
+-GSATT
+-GSATW
+-GSSMI
+-GAIREP
+-GPSEUD—FILL
+-S_U_RAOB
+-S_U_PIBAL
+-S_U_SFCM
+-S_U_SATT
+-S_U_ACARS
+-S_U_AMDAR
+-S_U_AIREP
+-S_U_SATW
+-S_U_SATWV
+-S_U_SSMI
+-S_U_SCAT
+-S_U_PSEUD
+-OBNRM
+-CMOV
+-COBSORT
+-DCOVARF-+-X1X2
+-GCIRCV
+-GFUNCE0
+-XFORMV
+-Z1Z2
+-ZFUNCE
+-ZMOD
+-ARRAYMF
+-MOV
+-XPAND
+-P_MAX
+-VCHLSK—XLFIL
+-CHSLV
+-OBREJ-+-XLIREJ-+-NCHSLV -+-CHSLV
+-TRANSP
+-INVT-+-INVTX
+-AMULT
+-NCHSLV
+-TRANSP
+-SORTA2
+-IXLAMD —INVT
+-CHSLV
+-UPREJ-+-ADDREJ
+-UACARS
+-UAMDAR
+-UAIREP
+-UPIBAL
+-URAOB
+-USATT
+-USATW
+-USATWV
+-USFCM
+-USSMI
+-USCAT
+-UPSEUD
+-COBREJ—IXLAMD
+-GPANF—PCOVARF -+-GCIRCV
+-GFUNCV0
+-XFORMV
+-ZFUNCP
+-ZMODP
+-ARRAYBF
+-ANLNRM
+-PX_SECT
+-VAR_BAL-+-GIPLVL
+-INTRP_H_U
+-INTRP_H_V
+-S_WND
+-GSTR_BAL
+-NL_BEQ
+-PX_SECT
+-QPRNTN
+-UNS_WND-+-FILL
+-INTRP_U_H
+-INTRP_V_H
+-OB_ST-+-GETF0
+-G_USET
+-VGRID
+-DTGNUM
+-DRAOB
+-DSFCM
+-DSATT
+-DSATW
+-DSATWV
+-DPIBAL
+-DACARS
+-DAMDAR
+-DSSMI
+-DSCAT
+-DAIREP
+-DPSEUD
+-QPRAOB-+-GIPLVL
+-UVCONV
+-QPPSEUD-+-GIPLVL
+-UVCONV
+-RRAOBT_NEW-+-FILL
+-IFILL
+-GTOL
+-DATRCD_NEW
+-LL_IJ
+-GETVAR
+-DF_UV
+-SORTQT
+-RRAOBT-+-FILL
+-IFILL

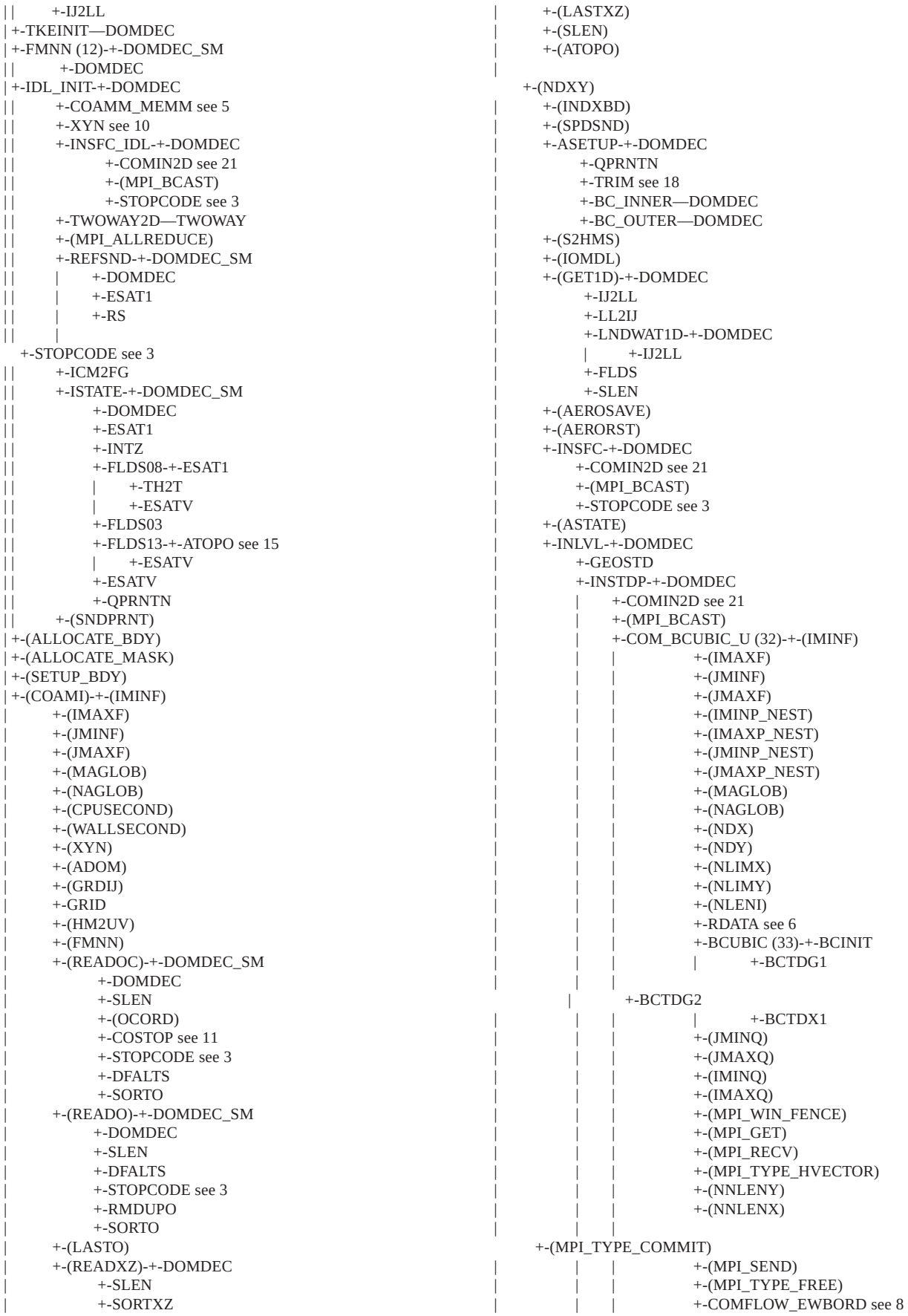
```
|     |      +-GTOL
|     |      +-DATRCD
|     |      +-DATRCD_NEW
|     |      +-LL_IJ
|     |      +-GETVAR
|     |      +-DF_UV
|     |      +-SORTQT
|     +-TANAL—FLD_OB
|     +-QANAL—FLD_OB
|     +-TANALS-+-DTGMOD
|     |      +-DTGNUM
|     |      +-INSSC
|     |      +-QPRNTN
|     |      +-FLD_OB
|     +-SVRAOB
|     +-UPRAOB-+-SLEN
|     |      +-(LOPEN)
|     |      +-COSTOP
|     |      +-(LCLOSE)
|     +-UPPSEUD-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPACARS-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPAMDAR-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPAIREP-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPPIBAL-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPSATT-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPSATW-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPSATWV-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPSFCM-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPSFCP-+-SLEN
|     |       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPSSMI-+-SLEN
|    £|       +-(LOPEN)
|     |       +-COSTOP
|     |       +-(LCLOSE)
|     +-UPSCAT-+-SLEN
|             +-(LOPEN)
|             +-COSTOP
|             +-(LCLOSE)
+-ANLFLD-+-CTEMPZ
|     +-GEOSTD
|     +-FILT9
+-NETINC—CM2FM
+-PRINTN—QPRNTN
+-PRINTF—QPRNTN
+-SUBDVFG
+-OUTANL—WDATA
+-OUTINC—WDATA
+-GETBDY-+-GEOSTD
|     +-STDP2Z-+-DOMDEC_SM
|             +-TERRPR-+-DOMDEC_SM
|             |      +-S2SINT    -+-DOMDEC_SM
|             +-QPRNTN
|             +-CTEMPZ
|             +-Z2ZINN-+-DOMDEC_SM
|             +-Z2ZIN-+-DOMDEC_SM
|             +-PSTD
|             +-PSTD1
|             +-S2SINT
|             +-VARADJ-+-DOMDEC_SM
|             |      +-GAUSSE-+-DOMDEC_SM
|             +-QSAT
+-TENDBD-+-S2HMS
|     +-WDATA
+-(DBSTOP)
```

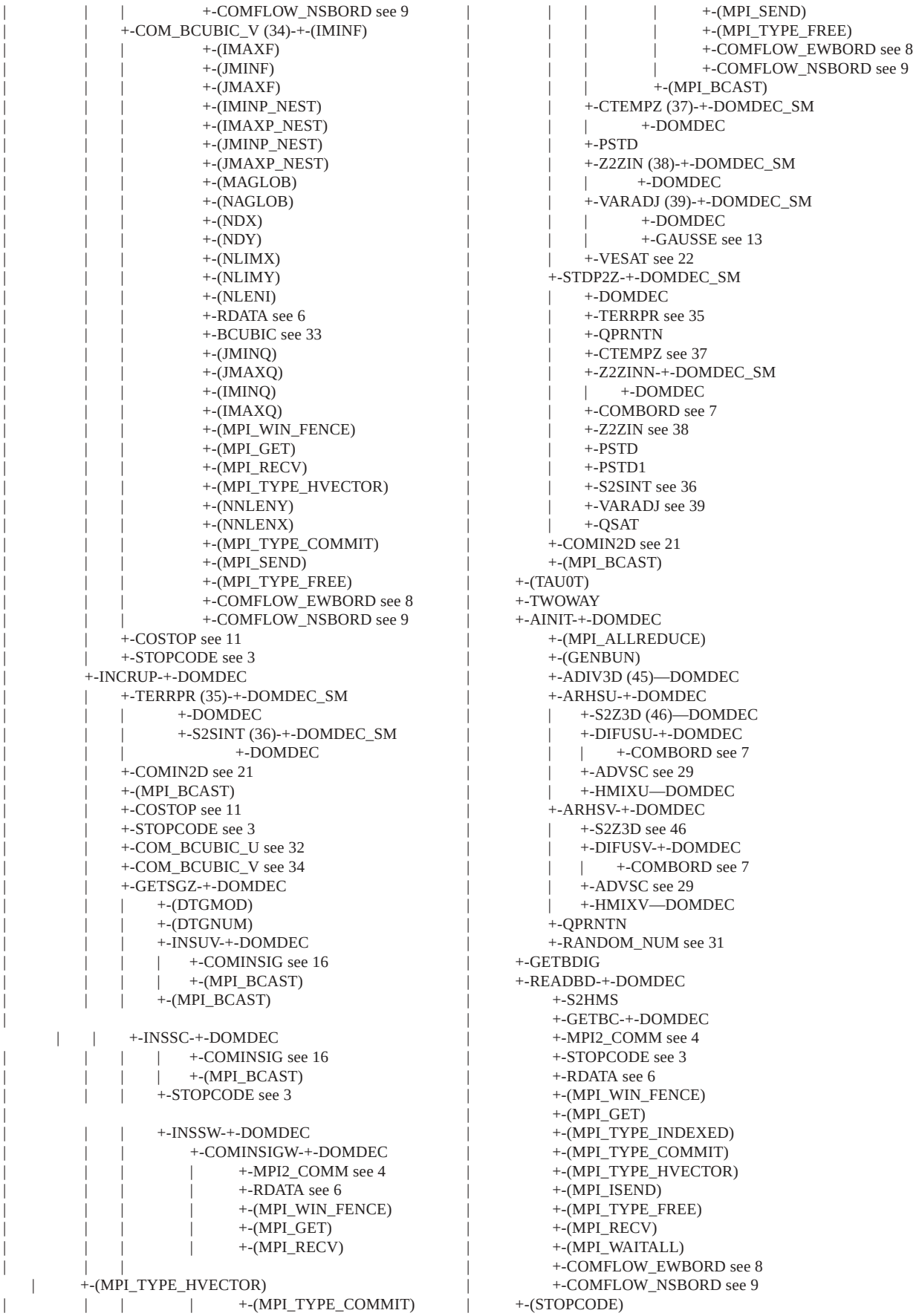
Appendix F

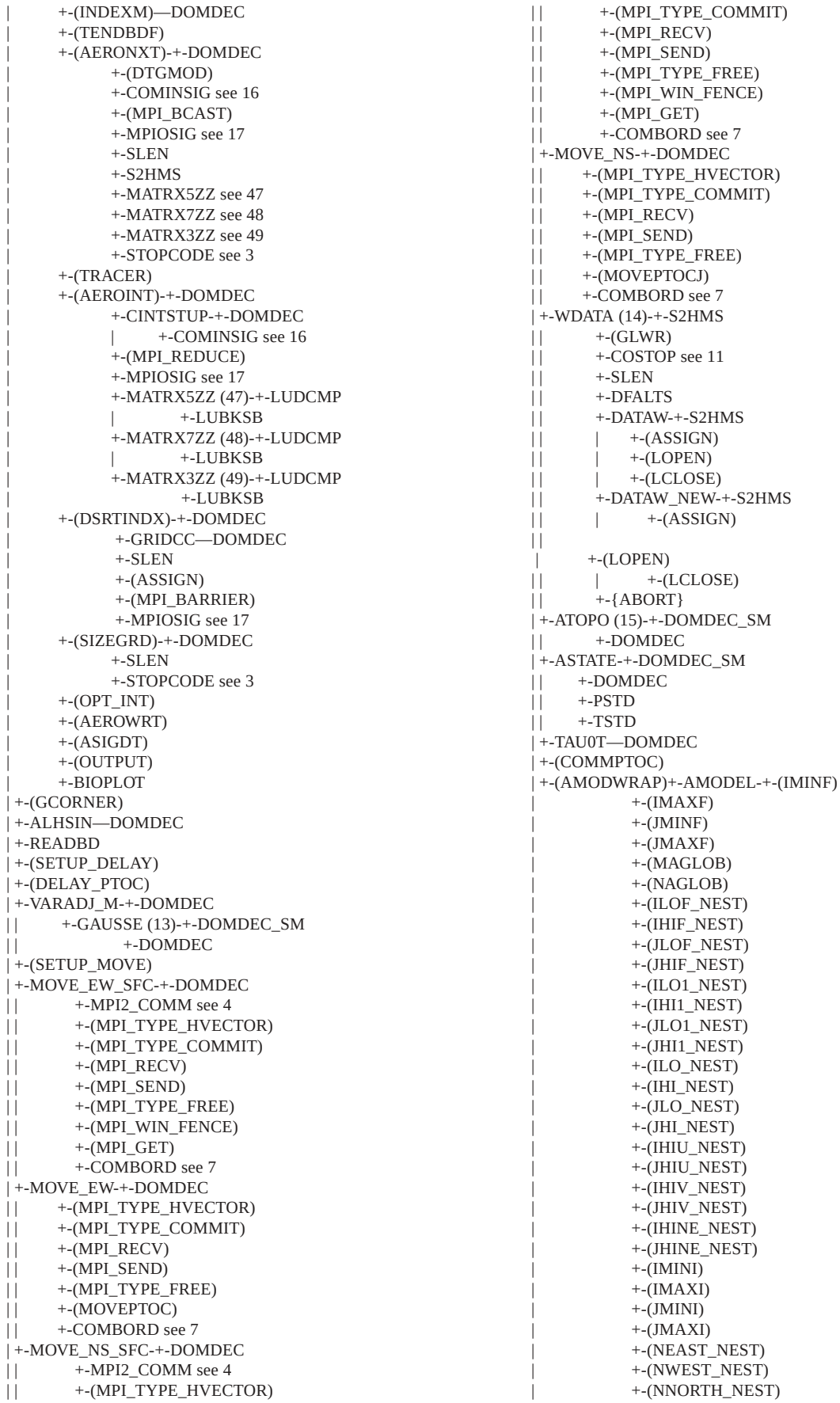
COAMPS Atmosphere Forecast Flow Chart

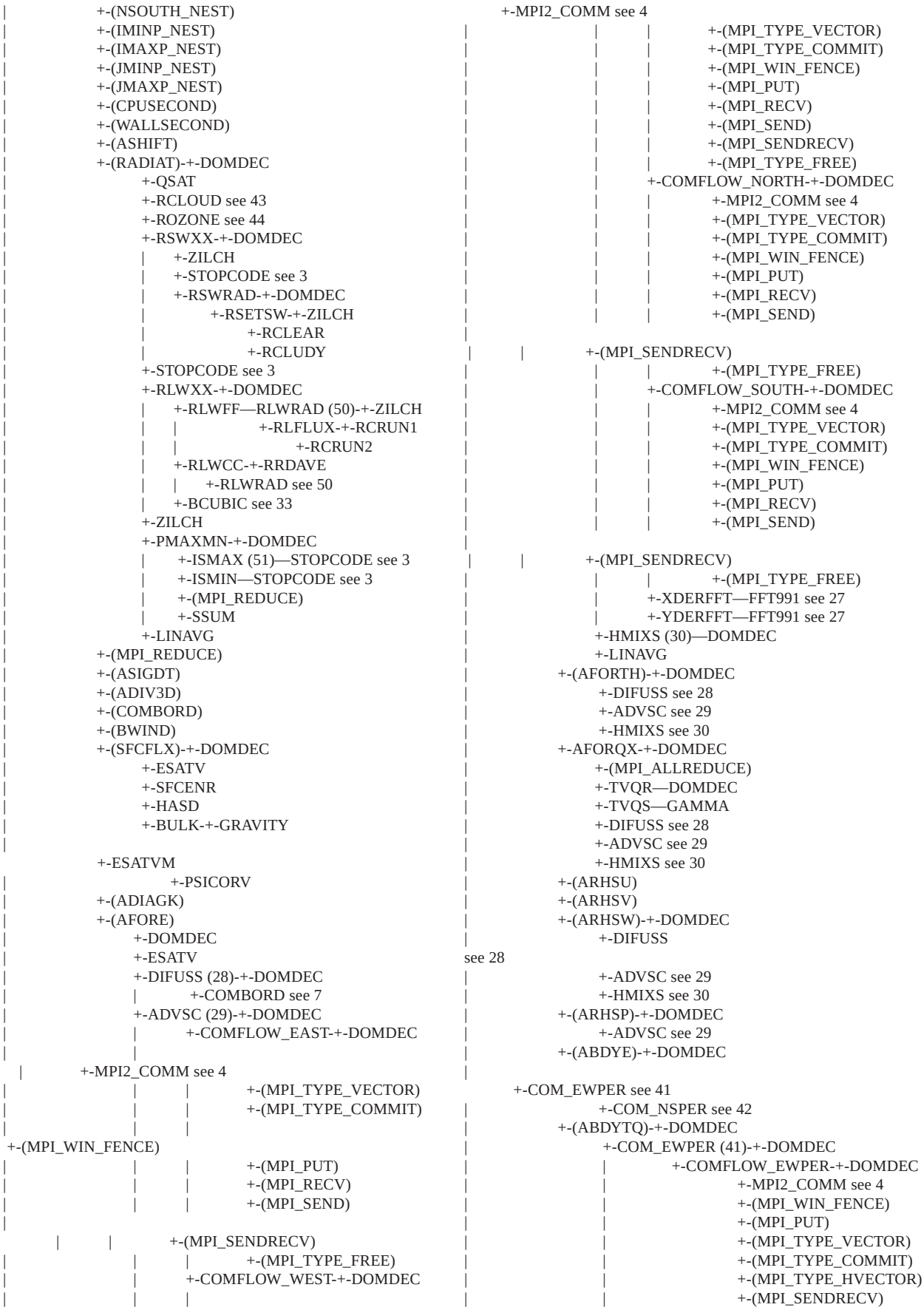
This is a primary tree starting at the program ‘ATMOS_FORECAST’.

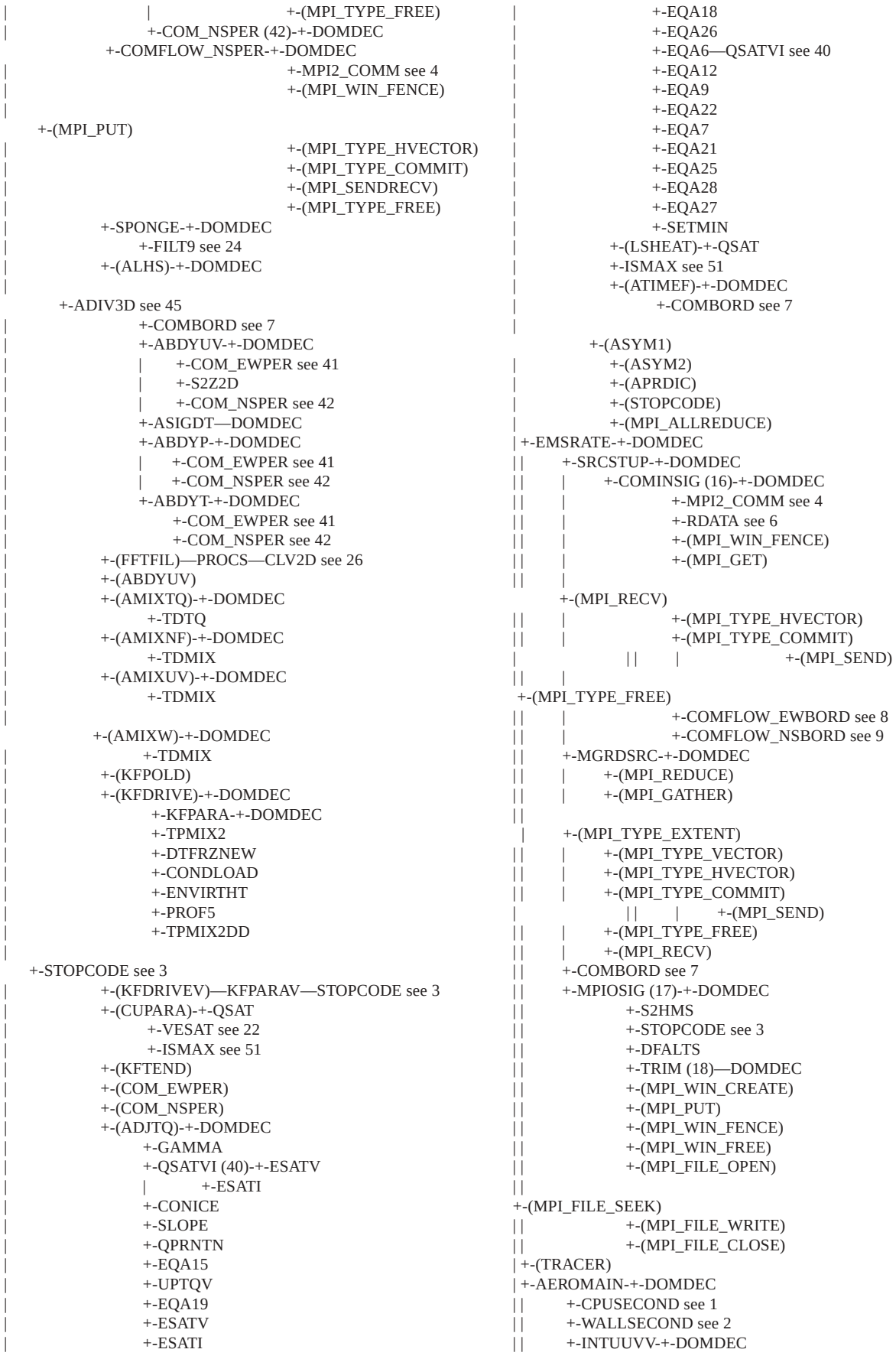
```
ATMOS_FORECAST
+-(MPI_INIT)
+-(MPI_COMM_RANK)
+-(MPI_COMM_SIZE)
+-CPUSECOND (1)-+-(SECOND)
|           +-(ETIME_)
|           +-{ETIME}
+-WALLSECOND (2)—(GET_WALLT)
+-STOPCODE (3)-+-(MPI_ABORT)
|          +-(MPI_BARRIER)
+-(MPI_COMM_SPLIT)
+-(DTGOPS)
+-(DOMDEC_INIT)—DOMDEC_NEST—STOPCODE see 3
+-(ALLOCATE_MEMM)-+-STOPCODE see 3
           +-ALLOCATE_NEST—STOPCODE see 3
+-(MPI_REDUCE)
+-(NTOTAL)
+-(IWIDTH)
+-(JWIDTH)
+-COAMM
| +-DOMDEC
| +-MPI2_COMM (4)—DOMDEC
| +-CPUSECOND see 1
| +-WALLSECOND see 2
| +-STOPCODE see 3
| +-DOMDEC_INIT2
| +-MPI2_INIT-+-(MPI_TYPE_EXTENT)
| |         +-(MPI_WIN_CREATE)
| +-GRDIJ-+-DOMDEC_SM
| |     +-DOMDEC
| +-(ADOM)
| +-GRID
| +-CHEKTS-+-DOMDEC_SM
| |       +-DOMDEC
| |       +-COAMM_MEMM (5)—DOMDEC
| |       +-(MPI_ALLREDUCE)
| +-RDATA (6)-+-S2HMS
| |           +-(GLRD)
| |           +-SLEN
| |           +-DFALTS
| |           +-DATAR-+-S2HMS
| |           |       +-(ASSIGN)
| |           +-DATAR_NEW-+-S2HMS
| |           |           +-(ASSIGN)
| |           +-{ABORT}
| +-(DTGMOD)
| +-CHECKGEOG
| +-SLEN
| +-S2HMS
| +-HM2UV
| | +-DOMDEC_SM
| | +-DOMDEC
| | +-COMBORD (7)-+-DOMDEC
| |               +-COMFLOW_BORD-+-DOMDEC
| |                              +-COMFLOW_EWBORD (8)-+-DOMDEC
| |                              |                    +-MPI2_COMM see 4
| |                              |                    +-(MPI_WIN_FENCE)
| |                              |                    +-(MPI_PUT)
| |                              |                    +-(MPI_TYPE_VECTOR)
| |                              |                    +-(MPI_TYPE_COMMIT)
| |                              |                    +-(MPI_TYPE_HVECTOR)
| |                              |                    +-(MPI_RECV)
| |                              |                    +-(MPI_SEND)
| |                              |                    +-(MPI_SENDRECV)
| |                              |                    +-(MPI_TYPE_FREE)
| |                              +-COMFLOW_NSBORD (9)-+-DOMDEC
| |
                                                    +-MPI2_COMM see 4
| |                                                   +-(MPI_WIN_FENCE)
| |                                                   +-(MPI_PUT)
| |
 +-(MPI_TYPE_HVECTOR)
| |                                                   +-(MPI_TYPE_COMMIT)
| |                                                   +-(MPI_RECV)
| |                                                   +-(MPI_SEND)
| |
                                          +-(MPI_SENDRECV)
| |                                                   +-(MPI_TYPE_FREE)
| +-(STAGF)
| +-XYN (10)-+-DOMDEC_SM
| |          +-DOMDEC
| +-COSTOP (11)-+-DOMDEC_SM
| |             +-DOMDEC
| |             +-(DBSTOP)
| |             +-STOPCODE see 3
| +-LL2IJ
| +-(MPI_BCAST)
| +-DAYCON-+-DOMDEC
| |        +-DAYNUM
| +-ESAT1
| +-NSTLVLS-+-DOMDEC
| |         +-STOPCODE see 3
| +-SET99-+-FAX
| |       +-STOPCODE see 3
| |       +-FFTRIG
| +-AVGINIT
| +-LUTAB
| +-CHECKZ-+-DOMDEC_SM
| |        +-DOMDEC
| +-DZQRS—DOMDEC
| +-CHEKMN-+-DOMDEC_SM
| |        +-DOMDEC
| |        +-COSTOP see 11
| |        +-STOPCODE see 3
| +-ZEROBND—QPRNTN
| +-CHKISIS-+-(GGRD)
| |         +-COSTOP see 11
| |         +-(GETGEOM)
```


+-IJ2LL
+-TKEINIT—DOMDEC
+-FMNN (12)-+-DOMDEC_SM
+-DOMDEC
+-IDL_INIT-+-DOMDEC
+-COAMM_MEMM see 5
+-XYN see 10
+-INSFC_IDL-+-DOMDEC
+-COMIN2D see 21
+-(MPI_BCAST)
+-STOPCODE see 3
+-TWOWAY2D—TWOWAY
+-(MPI_ALLREDUCE)
+-REFSND-+-DOMDEC_SM
+-DOMDEC
+-ESAT1
+-RS
+-STOPCODE see 3
+-ICM2FG
+-ISTATE-+-DOMDEC_SM
+-DOMDEC
+-ESAT1
+-INTZ
+-FLDS08-+-ESAT1
+-TH2T
+-ESATV
+-FLDS03
+-FLDS13-+-ATOPO see 15
+-ESATV
+-ESATV
+-QPRNTN
+-(SNDPRNT)
+-(ALLOCATE_BDY)
+-(ALLOCATE_MASK)
+-(SETUP_BDY)
+-(COAMI)-+-(IMINF)
+-(IMAXF)
+-(JMINF)
+-(JMAXF)
+-(MAGLOB)
+-(NAGLOB)
+-(CPUSECOND)
+-(WALLSECOND)
+-(XYN)
+-(ADOM)
+-(GRDIJ)
+-GRID
+-(HM2UV)
+-(FMNN)
+-(READOC)-+-DOMDEC_SM
+-DOMDEC
+-SLEN
+-(OCORD)
+-COSTOP see 11
+-STOPCODE see 3
+-DFALTS
+-SORTO
+-(READO)-+-DOMDEC_SM
+-DOMDEC
+-SLEN
+-DFALTS
+-STOPCODE see 3
+-RMDUPO
+-SORTO
+-(LASTO)
+-(READXZ)-+-DOMDEC
+-SLEN
+-SORTXZ
+-(LASTXZ)
+-(SLEN)
+-(ATOPO)
+-(NDXY)
+-(INDXBD)
+-(SPDSND)
+-ASETUP-+-DOMDEC
+-QPRNTN
+-TRIM see 18
+-BC_INNER—DOMDEC
+-BC_OUTER—DOMDEC
+-(S2HMS)
+-(IOMDL)
+-(GET1D)-+-DOMDEC
+-IJ2LL
+-LL2IJ
+-LNDWAT1D-+-DOMDEC
+-IJ2LL
+-FLDS
+-SLEN
+-(AEROSAVE)
+-(AERORST)
+-INSFC-+-DOMDEC
+-COMIN2D see 21
+-(MPI_BCAST)
+-STOPCODE see 3
+-(ASTATE)
+-INLVL-+-DOMDEC
+-GEOSTD
+-INSTDP-+-DOMDEC
+-COMIN2D see 21
+-(MPI_BCAST)
+-COM_BCUBIC_U (32)-+-(IMINF)
+-(IMAXF)
+-(JMINF)
+-(JMAXF)
+-(IMINP_NEST)
+-(IMAXP_NEST)
+-(JMINP_NEST)
+-(JMAXP_NEST)
+-(MAGLOB)
+-(NAGLOB)
+-(NDX)
+-(NDY)
+-(NLIMX)
+-(NLIMY)
+-(NLENI)
+-RDATA see 6
+-BCUBIC (33)-+-BCINIT
+-BCTDG1
+-BCTDG2
+-BCTDX1
+-(JMINQ)
+-(JMAXQ)
+-(IMINQ)
+-(IMAXQ)
+-(MPI_WIN_FENCE)
+-(MPI_GET)
+-(MPI_RECV)
+-(MPI_TYPE_HVECTOR)
+-(NNLENY)
+-(NNLENX)
+-(MPI_TYPE_COMMIT)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-COMFLOW_EWBORD see 8


+-COMFLOW_NSBORD see 9
+-COM_BCUBIC_V (34)-+-(IMINF)
+-(IMAXF)
+-(JMINF)
+-(JMAXF)
+-(IMINP_NEST)
+-(IMAXP_NEST)
+-(JMINP_NEST)
+-(JMAXP_NEST)
+-(MAGLOB)
+-(NAGLOB)
+-(NDX)
+-(NDY)
+-(NLIMX)
+-(NLIMY)
+-(NLENI)
+-RDATA see 6
+-BCUBIC see 33
+-(JMINQ)
+-(JMAXQ)
+-(IMINQ)
+-(IMAXQ)
+-(MPI_WIN_FENCE)
+-(MPI_GET)
+-(MPI_RECV)
+-(MPI_TYPE_HVECTOR)
+-(NNLENY)
+-(NNLENX)
+-(MPI_TYPE_COMMIT)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-COMFLOW_EWBORD see 8
+-COMFLOW_NSBORD see 9
+-COSTOP see 11
+-STOPCODE see 3
+-INCRUP-+-DOMDEC
+-TERRPR (35)-+-DOMDEC_SM
+-DOMDEC
+-S2SINT (36)-+-DOMDEC_SM
+-DOMDEC
+-COMIN2D see 21
+-(MPI_BCAST)
+-COSTOP see 11
+-STOPCODE see 3
+-COM_BCUBIC_U see 32
+-COM_BCUBIC_V see 34
+-GETSGZ-+-DOMDEC
+-(DTGMOD)
+-(DTGNUM)
+-INSUV-+-DOMDEC
+-COMINSIG see 16
+-(MPI_BCAST)
+-(MPI_BCAST)
+-INSSC-+-DOMDEC
+-COMINSIG see 16
+-(MPI_BCAST)
+-STOPCODE see 3
+-INSSW-+-DOMDEC
+-COMINSIGW-+-DOMDEC
+-MPI2_COMM see 4
+-RDATA see 6
+-(MPI_WIN_FENCE)
+-(MPI_GET)
+-(MPI_RECV)
+-(MPI_TYPE_HVECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-COMFLOW_EWBORD see 8
+-COMFLOW_NSBORD see 9
+-(MPI_BCAST)
+-CTEMPZ (37)-+-DOMDEC_SM
+-DOMDEC
+-PSTD
+-Z2ZIN (38)-+-DOMDEC_SM
+-DOMDEC
+-VARADJ (39)-+-DOMDEC_SM
+-DOMDEC
+-GAUSSE see 13
+-VESAT see 22
+-STDP2Z-+-DOMDEC_SM
+-DOMDEC
+-TERRPR see 35
+-QPRNTN
+-CTEMPZ see 37
+-Z2ZINN-+-DOMDEC_SM
+-DOMDEC
+-COMBORD see 7
+-Z2ZIN see 38
+-PSTD
+-PSTD1
+-S2SINT see 36
+-VARADJ see 39
+-QSAT
+-COMIN2D see 21
+-(MPI_BCAST)
+-(TAU0T)
+-TWOWAY
+-AINIT-+-DOMDEC
+-(MPI_ALLREDUCE)
+-(GENBUN)
+-ADIV3D (45)—DOMDEC
+-ARHSU-+-DOMDEC
+-S2Z3D (46)—DOMDEC
+-DIFUSU-+-DOMDEC
+-COMBORD see 7
+-ADVSC see 29
+-HMIXU—DOMDEC
+-ARHSV-+-DOMDEC
+-S2Z3D see 46
+-DIFUSV-+-DOMDEC
+-COMBORD see 7
+-ADVSC see 29
+-HMIXV—DOMDEC
+-QPRNTN
+-RANDOM_NUM see 31
+-GETBDIG
+-READBD-+-DOMDEC
+-S2HMS
+-GETBC-+-DOMDEC
+-MPI2_COMM see 4
+-STOPCODE see 3
+-RDATA see 6
+-(MPI_WIN_FENCE)
+-(MPI_GET)
+-(MPI_TYPE_INDEXED)
+-(MPI_TYPE_COMMIT)
+-(MPI_TYPE_HVECTOR)
+-(MPI_ISEND)
+-(MPI_TYPE_FREE)
+-(MPI_RECV)
+-(MPI_WAITALL)
+-COMFLOW_EWBORD see 8
+-COMFLOW_NSBORD see 9
+-(STOPCODE)


+-(INDEXM)—DOMDEC
+-(TENDBDF)
+-(AERONXT)-+-DOMDEC
+-(DTGMOD)
+-COMINSIG see 16
+-(MPI_BCAST)
+-MPIOSIG see 17
+-SLEN
+-S2HMS
+-MATRX5ZZ see 47
+-MATRX7ZZ see 48
+-MATRX3ZZ see 49
+-STOPCODE see 3
+-(TRACER)
+-(AEROINT)-+-DOMDEC
+-CINTSTUP-+-DOMDEC
+-COMINSIG see 16
+-(MPI_REDUCE)
+-MPIOSIG see 17
+-MATRX5ZZ (47)-+-LUDCMP
+-LUBKSB
+-MATRX7ZZ (48)-+-LUDCMP
+-LUBKSB
+-MATRX3ZZ (49)-+-LUDCMP
+-LUBKSB
+-(DSRTINDX)-+-DOMDEC
+-GRIDCC—DOMDEC
+-SLEN
+-(ASSIGN)
+-(MPI_BARRIER)
+-MPIOSIG see 17
+-(SIZEGRD)-+-DOMDEC
+-SLEN
+-STOPCODE see 3
+-(OPT_INT)
+-(AEROWRT)
+-(ASIGDT)
+-(OUTPUT)
+-BIOPLOT
+-(GCORNER)
+-ALHSIN—DOMDEC
+-READBD
+-(SETUP_DELAY)
+-(DELAY_PTOC)
+-VARADJ_M-+-DOMDEC
+-GAUSSE (13)-+-DOMDEC_SM
+-DOMDEC
+-(SETUP_MOVE)
+-MOVE_EW_SFC-+-DOMDEC
+-MPI2_COMM see 4
+-(MPI_TYPE_HVECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_RECV)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-(MPI_WIN_FENCE)
+-(MPI_GET)
+-COMBORD see 7
+-MOVE_EW-+-DOMDEC
+-(MPI_TYPE_HVECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_RECV)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-(MOVEPTOC)
+-COMBORD see 7
+-MOVE_NS_SFC-+-DOMDEC
+-MPI2_COMM see 4
+-(MPI_TYPE_HVECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_RECV)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-(MPI_WIN_FENCE)
+-(MPI_GET)
+-COMBORD see 7
+-MOVE_NS-+-DOMDEC
+-(MPI_TYPE_HVECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_RECV)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-(MOVEPTOCJ)
+-COMBORD see 7
+-WDATA (14)-+-S2HMS
+-(GLWR)
+-COSTOP see 11
+-SLEN
+-DFALTS
+-DATAW-+-S2HMS
+-(ASSIGN)
+-(LOPEN)
+-(LCLOSE)
+-DATAW_NEW-+-S2HMS
+-(ASSIGN)
+-(LOPEN)
+-(LCLOSE)
+-{ABORT}
+-ATOPO (15)-+-DOMDEC_SM
+-DOMDEC
+-ASTATE-+-DOMDEC_SM
+-DOMDEC
+-PSTD
+-TSTD
+-TAU0T—DOMDEC
+-(COMMPTOC)
+-(AMODWRAP)+-AMODEL-+-(IMINF)
+-(IMAXF)
+-(JMINF)
+-(JMAXF)
+-(MAGLOB)
+-(NAGLOB)
+-(ILOF_NEST)
+-(IHIF_NEST)
+-(JLOF_NEST)
+-(JHIF_NEST)
+-(ILO1_NEST)
+-(IHI1_NEST)
+-(JLO1_NEST)
+-(JHI1_NEST)
+-(ILO_NEST)
+-(IHI_NEST)
+-(JLO_NEST)
+-(JHI_NEST)
+-(IHIU_NEST)
+-(JHIU_NEST)
+-(IHIV_NEST)
+-(JHIV_NEST)
+-(IHINE_NEST)
+-(JHINE_NEST)
+-(IMINI)
+-(IMAXI)
+-(JMINI)
+-(JMAXI)
+-(NEAST_NEST)
+-(NWEST_NEST)
+-(NNORTH_NEST)


+-(NSOUTH_NEST)
+-(IMINP_NEST)
+-(IMAXP_NEST)
+-(JMINP_NEST)
+-(JMAXP_NEST)
+-(CPUSECOND)
+-(WALLSECOND)
+-(ASHIFT)
+-(RADIAT)-+-DOMDEC
+-QSAT
+-RCLOUD see 43
+-ROZONE see 44
+-RSWXX-+-DOMDEC
+-ZILCH
+-STOPCODE see 3
+-RSWRAD-+-DOMDEC
+-RSETSW-+-ZILCH
+-RCLEAR
+-RCLUDY
+-STOPCODE see 3
+-RLWXX-+-DOMDEC
+-RLWFF—RLWRAD (50)-+-ZILCH
+-RLFLUX-+-RCRUN1
+-RCRUN2
+-RLWCC-+-RRDAVE
+-RLWRAD see 50
+-BCUBIC see 33
+-ZILCH
+-PMAXMN-+-DOMDEC
+-ISMAX (51)—STOPCODE see 3
+-ISMIN—STOPCODE see 3
+-(MPI_REDUCE)
+-SSUM
+-LINAVG
+-(MPI_REDUCE)
+-(ASIGDT)
+-(ADIV3D)
+-(COMBORD)
+-(BWIND)
+-(SFCFLX)-+-DOMDEC
+-ESATV
+-SFCENR
+-HASD
+-BULK-+-GRAVITY
+-ESATVM
+-PSICORV
+-(ADIAGK)
+-(AFORE)
+-DOMDEC
+-ESATV
+-DIFUSS (28)-+-DOMDEC
+-COMBORD see 7
+-ADVSC (29)-+-DOMDEC
+-COMFLOW_EAST-+-DOMDEC
+-MPI2_COMM see 4
+-(MPI_TYPE_VECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_WIN_FENCE)
+-(MPI_PUT)
+-(MPI_RECV)
+-(MPI_SEND)
+-(MPI_SENDRECV)
+-(MPI_TYPE_FREE)
+-COMFLOW_WEST-+-DOMDEC
+-MPI2_COMM see 4
+-(MPI_TYPE_VECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_WIN_FENCE)
+-(MPI_PUT)
+-(MPI_RECV)
+-(MPI_SEND)
+-(MPI_SENDRECV)
+-(MPI_TYPE_FREE)
+-COMFLOW_NORTH-+-DOMDEC
+-MPI2_COMM see 4
+-(MPI_TYPE_VECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_WIN_FENCE)
+-(MPI_PUT)
+-(MPI_RECV)
+-(MPI_SEND)
+-(MPI_SENDRECV)
+-(MPI_TYPE_FREE)
+-COMFLOW_SOUTH-+-DOMDEC
+-MPI2_COMM see 4
+-(MPI_TYPE_VECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_WIN_FENCE)
+-(MPI_PUT)
+-(MPI_RECV)
+-(MPI_SEND)
+-(MPI_SENDRECV)
+-(MPI_TYPE_FREE)
+-XDERFFT—FFT991 see 27
+-YDERFFT—FFT991 see 27
+-HMIXS (30)—DOMDEC
+-LINAVG
+-(AFORTH)-+-DOMDEC
+-DIFUSS see 28
+-ADVSC see 29
+-HMIXS see 30
+-AFORQX-+-DOMDEC
+-(MPI_ALLREDUCE)
+-TVQR—DOMDEC
+-TVQS—GAMMA
+-DIFUSS see 28
+-ADVSC see 29
+-HMIXS see 30
+-(ARHSU)
+-(ARHSV)
+-(ARHSW)-+-DOMDEC
+-DIFUSS
see 28
+-ADVSC see 29
+-HMIXS see 30
+-(ARHSP)-+-DOMDEC
+-ADVSC see 29
+-(ABDYE)-+-DOMDEC
+-COM_EWPER see 41
+-COM_NSPER see 42
+-(ABDYTQ)-+-DOMDEC
+-COM_EWPER (41)-+-DOMDEC
+-COMFLOW_EWPER-+-DOMDEC
+-MPI2_COMM see 4
+-(MPI_WIN_FENCE)
+-(MPI_PUT)
+-(MPI_TYPE_VECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_TYPE_HVECTOR)
+-(MPI_SENDRECV)


+-(MPI_TYPE_FREE)
+-COM_NSPER (42)-+-DOMDEC
+-COMFLOW_NSPER-+-DOMDEC
+-MPI2_COMM see 4
+-(MPI_WIN_FENCE)
+-(MPI_PUT)
+-(MPI_TYPE_HVECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_SENDRECV)
+-(MPI_TYPE_FREE)
+-SPONGE-+-DOMDEC
+-FILT9 see 24
+-(ALHS)-+-DOMDEC
+-ADIV3D see 45
+-COMBORD see 7
+-ABDYUV-+-DOMDEC
+-COM_EWPER see 41
+-S2Z2D
+-COM_NSPER see 42
+-ASIGDT—DOMDEC
+-ABDYP-+-DOMDEC
+-COM_EWPER see 41
+-COM_NSPER see 42
+-ABDYT-+-DOMDEC
+-COM_EWPER see 41
+-COM_NSPER see 42
+-(FFTFIL)—PROCS—CLV2D see 26
+-(ABDYUV)
+-(AMIXTQ)-+-DOMDEC
+-TDTQ
+-(AMIXNF)-+-DOMDEC
+-TDMIX
+-(AMIXUV)-+-DOMDEC
+-TDMIX
+-(AMIXW)-+-DOMDEC
+-TDMIX
+-(KFPOLD)
+-(KFDRIVE)-+-DOMDEC
+-KFPARA-+-DOMDEC
+-TPMIX2
+-DTFRZNEW
+-CONDLOAD
+-ENVIRTHT
+-PROF5
+-TPMIX2DD
+-STOPCODE see 3
+-(KFDRIVEV)—KFPARAV—STOPCODE see 3
+-(CUPARA)-+-QSAT
+-VESAT see 22
+-ISMAX see 51
+-(KFTEND)
+-(COM_EWPER)
+-(COM_NSPER)
+-(ADJTQ)-+-DOMDEC
+-GAMMA
+-QSATVI (40)-+-ESATV
+-ESATI
+-CONICE
+-SLOPE
+-QPRNTN
+-EQA15
+-UPTQV
+-EQA19
+-ESATV
+-ESATI
+-EQA18
+-EQA26
+-EQA6—QSATVI see 40
+-EQA12
+-EQA9
+-EQA22
+-EQA7
+-EQA21
+-EQA25
+-EQA28
+-EQA27
+-SETMIN
+-(LSHEAT)-+-QSAT
+-ISMAX see 51
+-(ATIMEF)-+-DOMDEC
+-COMBORD see 7
+-(ASYM1)
+-(ASYM2)
+-(APRDIC)
+-(STOPCODE)
+-(MPI_ALLREDUCE)
+-EMSRATE-+-DOMDEC
+-SRCSTUP-+-DOMDEC
+-COMINSIG (16)-+-DOMDEC
+-MPI2_COMM see 4
+-RDATA see 6
+-(MPI_WIN_FENCE)
+-(MPI_GET)
+-(MPI_RECV)
+-(MPI_TYPE_HVECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-COMFLOW_EWBORD see 8
+-COMFLOW_NSBORD see 9
+-MGRDSRC-+-DOMDEC
+-(MPI_REDUCE)
+-(MPI_GATHER)
+-(MPI_TYPE_EXTENT)
+-(MPI_TYPE_VECTOR)
+-(MPI_TYPE_HVECTOR)
+-(MPI_TYPE_COMMIT)
+-(MPI_SEND)
+-(MPI_TYPE_FREE)
+-(MPI_RECV)
+-COMBORD see 7
+-MPIOSIG (17)-+-DOMDEC
+-S2HMS
+-STOPCODE see 3
+-DFALTS
+-TRIM (18)—DOMDEC
+-(MPI_WIN_CREATE)
+-(MPI_PUT)
+-(MPI_WIN_FENCE)
+-(MPI_WIN_FREE)
+-(MPI_FILE_OPEN)
+-(MPI_FILE_SEEK)
+-(MPI_FILE_WRITE)
+-(MPI_FILE_CLOSE)
+-(TRACER)
+-AEROMAIN-+-DOMDEC
+-CPUSECOND see 1
+-WALLSECOND see 2
+-INTUUVV-+-DOMDEC

```
||       |       +-COMBORD see 7
||       +-SRCPNT—DOMDEC
||       +-DSRTSRC-+-DOMDEC
||       |       +-COMBORD see 7
||       +-MIXSCL-+-DOMDEC
||       |       +-DIFFUSZ-+-DOMDEC
||       |       |       +-TRIDAG-+-DOMDEC
||       |       |       |       +-STOPCODE see 3
||       |       |       +-(MPI_REDUCE)
||       |       |       +-COMBORD see 7
||       |       +-DIFFUSX-+-DOMDEC
||       |       |       +-(MPI_REDUCE)
||       |       |       +-COMBORD see 7
||       |       +-DIFFUSY-+-DOMDEC
||       |               +-(MPI_REDUCE)
||       |               +-COMBORD see 7
||       +-SEDIMENT-+-DOMDEC
||       |        +-COMBORD see 7
||       +-FLUXADV-+-DOMDEC
||       |       +-XFLUXM-+-DOMDEC
||       |       |       +-POLYN5X
||       |       |       +-COMFLOW_EWBORD see 8
||       |       |       +-POLYN7X
||       |       |       +-POLYN3X
||       |       |
    +-BOTT5XY
||       |       |       +-BOTT7XY
||       |       |       +-BOTT3XY
||       |       |       +-(MPI_REDUCE)
||       |       |       +-COMBORD see 7
||       |       +-YFLUXM-+-DOMDEC
||       |       |       +-POLYN5Y
||       |       |       +-COMFLOW_NSBORD see 9
||       |       |       +-POLYN7Y
||       |       |       +-POLYN3Y
||       |       |       +-BOTT5XY
||       |       |       +-BOTT7XY
||       |       |       +-BOTT3XY
||       |       |       +-(MPI_REDUCE)
||       |       |       +-COMBORD see 7
||       |       +-ZFLUXM-+-DOMDEC
||       |               +-BOTT5ZZ—POLYN5ZZ
||       |               +-BOTT7ZZ—POLYN7ZZ
||       |               +-BOTT3ZZ—POLYN3ZZ
||       |               +-(MPI_REDUCE)
||       |               +-COMBORD see 7
||       +-DRYDEPST-+-DOMDEC
||       |        +-(MPI_REDUCE)
||       |        +-COMBORD see 7
||       +-WETDEPST-+-DOMDEC
||       |        +-(MPI_REDUCE)
||       |        +-COMBORD see 7
||       +-BUDGET-+-DOMDEC
||              +-(MPI_REDUCE)
|+-BUNDRYCC-+-DOMDEC
||       +-CPUSECOND see 1
||       +-WALLSECOND see 2
||       +-STOPCODE see 3
||       +-(MPI_GATHER)
||       +-(MPI_TYPE_VECTOR)
||       +-(MPI_TYPE_COMMIT)
||       +-(MPI_SEND)
||       +-(MPI_TYPE_FREE)
||       +-(MPI_RECV)
||       +-(MPI_BARRIER)
|+-FEEDBKCC-+-DOMDEC
||       +-CPUSECOND see 1
||       +-WALLSECOND see 2
||       +-STOPCODE see 3
||       +-(MPI_GATHER)
||       +-(MPI_TYPE_VECTOR)
||       +-(MPI_TYPE_COMMIT)
||       +-(MPI_SEND)
||       +-(MPI_TYPE_FREE)
||       +-(MPI_RECV)
||       +-(MPI_BARRIER)
||       +-(MPI_BCAST)
||       +-COMBORD see 7
|+-FEEDBK
|+-OUTPUT-+-COAMM_MEMM see 5
||      +-IOSIG-+-DOMDEC
||      |     +-MPIOSIG see 17
||      +-OUTSFC-+-DOMDEC
||      |      +-SLEN
||      |      +-(DTGNUM)
||      |      +-MPIBOUND2 (19)—DOMDEC
||      |      +-UVG2UV (20)—DOMDEC
||      |      +-MPIBOUND1—DOMDEC
||      |      +-MPIOSIG see 17
||      |      +-COMIN2D (21)-+-DOMDEC
||      |      |            +-MPI2_COMM see 4
||      |      |            +-RDATA see 6
||      |      |            +-(MPI_WIN_FENCE)
||      |      |            +-(MPI_GET)
||      |      |            +-(MPI_RECV)
||      |      |            +-(MPI_TYPE_HVECTOR)
||      |      |            +-(MPI_TYPE_COMMIT)
||      |      |            +-(MPI_SEND)
||      |      |            +-(MPI_TYPE_FREE)
||      |      |            +-COMFLOW_EWBORD see 8
||      |      |            +-COMFLOW_NSBORD see 9
||      |      +-(MPI_BCAST)
||      |      +-STOPCODE see 3
||      |      +-VESAT (22)-+-DOMDEC_SM
||      |      |          +-DOMDEC
||      |      +-TROPPR-+-DOMDEC
||      |             +-S2PINT (23)—DOMDEC
||      |             +-FILT9 (24)-+-DOMDEC_SM
||      |             |          +-DOMDEC
||      |             |          +-COMBORD see 7
||      |             +-MPIBOUND2 see 19
||      +-AOUTP-+-DOMDEC
||      |      +-FILT9 see 24
||      |      +-MPIOSIG see 17
||      |      +-MPICOL-+-DOMDEC
||      |      |       +-MPI2_COMM see 4
||      |      |       +-(MPI_WIN_FENCE)
||      |      |       +-(MPI_PUT)
||      |      |       +-(MPI_TYPE_HVECTOR)
||      |      |       +-(MPI_TYPE_COMMIT)
||      |      |       +-(MPI_SEND)
||      |      |       +-(MPI_TYPE_FREE)
||      |      |       +-(MPI_RECV)
||      |      +-SLPRELAX
||      |      +-MPIDIS-+-DOMDEC
||      |      |       +-(MPI_RECV)
||      |      |       +-(MPI_TYPE_HVECTOR)
||      |      |       +-(MPI_TYPE_COMMIT)
||      |      |       +-(MPI_SEND)
||      |      |       +-(MPI_TYPE_FREE)
||      |      |       +-COMFLOW_EWBORD see 8
||      |      |       +-COMFLOW_NSBORD see 9
||      |      +-GEOSTD
||      |      +-TTSTD
||      |      +-S2PINT see 23
||      |      +-STD_PLVL
||      |      +-VESAT see 22
||      |      +-MPIBOUND2 see 19
```

```
||      |      +-UVG2UV see 20
||      +-AOUTZ-+-DOMDEC
||      |      +-DMDZ-+-DOMDEC
||      |      |      +-Z2ZINT (25)—DOMDEC
||      |      +-MPIOSIG see 17
||      |      +-EVPDUCT-+-DOMDEC
||      |      |        +-(DTGMOD)
||      |      |        +-COMIN2D see 21
||      |      |        +-(MPI_BCAST)
||      |      |        +-VESAT see 22
||      |      +-EM—DOMDEC
||      |      +-EMDUCT—DOMDEC
||      |      +-Z2ZINT see 25
||      |      +-VESAT see 22
||      |      +-MPIBOUND2 see 19
||      |      +-UVG2UV see 20
||      +-AOUTSIG-+-DOMDEC
||      |       +-VESAT see 22
||      |       +-MPIOSIG see 17
||      +-LASTO
|+-AEROWRT-+-DOMDEC
||       +-(MPI_REDUCE)
||       +-MPIOSIG see 17
||       +-SLEN
||       +-S2HMS
||       +-OPT_DEP—DOMDEC
||       +-STOPCODE see 3
|+-BIOPLOT
|+-IOMDL-+-COAMM_MEMM see 5
||      +-IOZAVG-+-S2HMS
||      |       +-SLEN
||      |       +-STOPCODE see 3
||      +-(IOSFC0)
||      +-(IOSFCT)
||      +-IOATMT-+-S2HMS
||      |       +-SLEN
||      |       +-COSTOP see 11
||      |       +-STOPCODE see 3
||      +-FMNN see 12
||      +-ATOPO see 15
||      +-INDXBD—DOMDEC
||      +-IOSAVE-+-COAMM_MEMM see 5
||               +-STOPCODE see 3
||               +-S2HMS
||               +-SLEN
||               +-(RENAMEW)
||               +-READ_RESTART-+-COAMM_MEMM see 5
||               |              +-(MPI_FILE_OPEN)
||               |              +-(MPI_FILE_SEEK)
||               |              +-(MPI_FILE_READ)
||               |              +-(MPI_FILE_CLOSE)
||               +-(MPI_BARRIER)
||               +-WRITE_RESTART-+-COAMM_MEMM see 5
||                               +-(MPI_FILE_OPEN)
||                               +-(MPI_FILE_SEEK)
||                               +-(MPI_FILE_WRITE)
||                               +-(MPI_FILE_CLOSE)
|+-AEROSAVE-+-DOMDEC
||        +-STOPCODE see 3
||        +-S2HMS
||        +-SLEN
||        +-(MPI_BCAST)
|+-FFTPRT—COSPECTRA-+-CLV2D (26)—FFT991 (27)-+-FFT99A
||                 |                          +-VPASSM
||                 |                          +-FFT99B
||                 +-STOPCODE see 3
|+-FFTVAL
|+-(MPI_REDUCE)
|+-(DBSTOP)
+-(DEALLOCATE_MEMM)
+-(MPI_BARRIER)
+-(MPI_FINALIZE)


AFORE-+-DOMDEC
    +-ESATV
    +-DIFUSS (28)-+-DOMDEC
    |             +-COMBORD see 7
    +-ADVSC (29)-+-DOMDEC
    |            +-COMFLOW_EAST-+-DOMDEC
    |            |              +-MPI2_COMM see 4
    |            |              +-(MPI_TYPE_VECTOR)
    |            |              +-(MPI_TYPE_COMMIT)
    |            |              +-(MPI_WIN_FENCE)
    |            |              +-(MPI_PUT)
    |            |              +-(MPI_RECV)
    |            |              +-(MPI_SEND)
    |            |              +-(MPI_SENDRECV)
    |            |              +-(MPI_TYPE_FREE)
    |            +-COMFLOW_WEST-+-DOMDEC
    |            |              +-MPI2_COMM see 4
    |            |              +-(MPI_TYPE_VECTOR)
    |            |              +-(MPI_TYPE_COMMIT)
    |            |              +-(MPI_WIN_FENCE)
    |            |              +-(MPI_PUT)
    |            |              +-(MPI_RECV)
    |            |              +-(MPI_SEND)
    |            |              +-(MPI_SENDRECV)
    |            |
    |            +-(MPI_TYPE_FREE)
    |            +-COMFLOW_NORTH-+-DOMDEC
    |            |               +-MPI2_COMM see 4
    |            |               +-(MPI_TYPE_VECTOR)
    |            |               +-(MPI_TYPE_COMMIT)
    |            |               +-(MPI_WIN_FENCE)
    |            |               +-(MPI_PUT)
    |            |               +-(MPI_RECV)
    |            |               +-(MPI_SEND)
    |            |               +-(MPI_SENDRECV)
    |            |
    |            +-(MPI_TYPE_FREE)
    |            +-COMFLOW_SOUTH-+-DOMDEC
    |            |               +-MPI2_COMM see 4
    |            |               +-(MPI_TYPE_VECTOR)
    |            |               +-(MPI_TYPE_COMMIT)
    |            |               +-(MPI_WIN_FENCE)
    |            |               +-(MPI_PUT)
    |            |               +-(MPI_RECV)
    |            |               +-(MPI_SEND)
    |            |               +-(MPI_SENDRECV)
    |            |
    |            +-(MPI_TYPE_FREE)
    |            +-XDERFFT—FFT991 see 27
    |            +-YDERFFT—FFT991 see 27
    +-HMIXS (30)—DOMDEC
    +-LINAVG
```

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
May 2003	Final	

4. TITLE AND SUBTITLE

COAMPS™ Version 3 Model Description - General Theory and Equations

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602435N, 0602704N

6. AUTHOR(S)

Sue Chen, James Cummings, James Doyle, Richard Hodur, Teddy Holt, Chi-Sann Liou, Ming Liu, James Ridout, Jerome Schmidt, and William Thompson (NRL)
Arthur Mirin and Gayle Sugiyama (LLNL)

5d. PROJECT NUMBER
X-2342

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Marine Meteorology Division
Monterey, CA 93943-5502

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/PU/7500--04-448

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Space and Naval Warfare System Command (PMW-185)
San Diego, CA 92110-3127

10. SPONSOR / MONITOR'S ACRONYM(S)

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This publication describes the general theory and equations used for Version 3 of the Coupled Ocean/Atmosphere Mesoscale Prediction System (COAMPS™). COAMPS™ is a state-of-the-art distributed memory, mesoscale weather and ocean prediction system developed by the Naval Research Laboratory. It can be run on many different computing platforms such as the IBM, SGI, DEC, and LINUX PC. The most significant changes from the previous shared-memory version (Version 2) is the use of the Message Passing Interface (MPI) and a two-dimensional domain decomposition technique to run on either massive parallel computers or smaller workstations. The Version 3 atmospheric forecast model has a much better (almost linear) run time, scaling up to 400 processors compared to Version 2. Other new improvements include: (a) 3D ocean MVOI, (b) moving and delayed nests, (c) a new bulk microphysics scheme, and (d) wet deposition of the aerosol/tracer module. Basic numerical weather prediction background and college-level math knowledge are preferable for the reader to gain the best understanding of this document. For COAMPS™ users who are already familiar with running the model, this document can be used as reference guide since it provides valuable detailed information such as the naming conventions of the equations used in the source code, common user-specified model setup variables, and source code flow charts.

15. SUBJECT TERMS

COAMPS™ Version 3 Model Description

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR	145	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Prepared by
Technical Information Services Branch
Naval Research Laboratory
Washington, DC

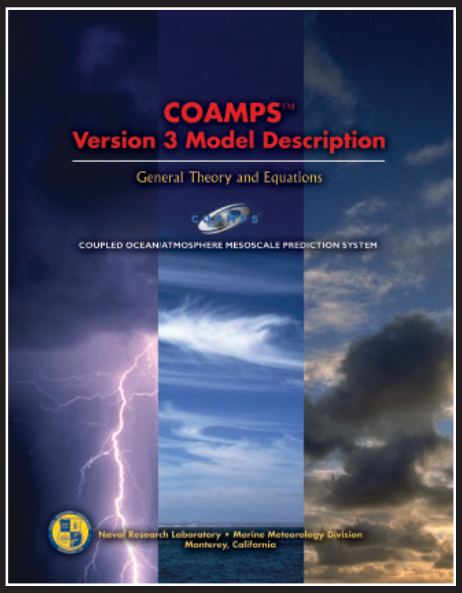
Editor
Maureen L. Long

Publication Design and Layout
Jan D. Morrow

Graphic Support
Donna Gloystein

Reviewed and Approved
NRL/PU/7500--03-448
May 2003

Philip E. Merilees
Superintendent
Marine Meteorology Division



http://www.nrlmry.navy.mil/projects/coamps